TAMPERE UNIVERSITY OF TECHNOLOGY

SAMI PIETIKÄINEN

REMOTE ISOBUS TELEMATICS IN AGRICULTURAL ENVIRONMENT

Master of Sciense Thesis

Examiner: Professor Hannu Koivisto
Examiner and topic approved in the Engineering Sciences Faculty Council meeting on 5th of March 2014.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Automation Technology
SAMI PIETIKÄINEN: Remote ISOBUS telematics in agricultural environment
Master of Science Thesis, 62 pages, 3 Appendix pages
June 2014
Major: Automation Software Engineering
Examiner: Professor Hannu Koivisto
Keywords: isobus, iso 11783, telematics, remote management

Many agricultural equipment manufacturers are mainly hardware providers, and the product life cycle services are limited to machine maintenance and spare parts. However, the modern wireless technologies makes it possible to access machines in their working environment, thus enabling many new applications. Accessing machines also requires accessing their internal communications network, which in agricultural machines is increasingly often ISOBUS.

The purpose of this thesis is to study the available telematics information and interfaces in an ISOBUS system, and also the applications this informations makes possible. Based on this information telematics functionalities are selected and implemented to Wapice Remote Management (WRM) system. Especially the standard manufacturer independent interfaces provided by ISO 11783 standard, in which the ISOBUS is based on, are studied.

The thesis is divided into three main parts. First, the current state-of-the-art is presented followed by usage examples for the telematics system which also highlights the main requirements for the system in whole. Thereafter, the second part introduces SAE J1939 protocol in which the ISOBUS is based on. The main concepts of the ISOBUS protocol are also introduced, and the ISOBUS is compared to other common CAN-based protocols. The results for the study of available information sources in an ISOBUS system as well as a general example reference architecture for the system are presented. In the third part the implementation of ISOBUS support in WRM system and the implementation of ISOBUS telematics functionalities is presented.

The results of this thesis suggest that the ISO 11783 standard provides many functionalities and interfaces that can be used to collect telematics as well as process information in a manufacturer independent way. This is a key factor when developing a generic system. The implementation for the WRM system also shows that it is possible to integrate ISOBUS telematics in an existing remote management system.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Automaatiotekniikan koulutusohjelma
SAMI PIETIKÄINEN: ISOBUS etätelematiikka maatalousympäristössä
Diplomityö, 62 sivua, 3 liitesivua
Kesäkuu 2014
Pääaine: Automaation ohjelmistotekniikka
Tarkastaja: Professori Hannu Koivisto
Avainsanat: isobus, iso 11783, telematiikka, etähallinta

Monet maatalouden työkonevalmistajat ovat pääasiassa laitetoimittajia, ja tuotteen elinaikaiset palvelut rajoittuvat huoltoon sekä varaosiin. Modernit langattomat teknologiat kuitenkin mahdollistaisivat pääsyn koneiden informaatiojärjestelmiin niiden työympäristössä, joka taas mahdollistaa monia uusia sovelluksia. Pääsy koneisiin vaatii myös pääsyä niiden sisäiseen kommunikointiväylään, joka maatalouskoneissa on yhä useammin ISOBUS.

Tämän opinnäytetyön tarkoituksena on tutkia, mitä telematiikkatietoa ISOBUS-järjestelmissä on saatavilla ja millaisia eri tyyppisiä sovelluksia tämä tieto mahdollistaa. Näiden tietojen perusteella valitaan ISOBUS-toiminnallisuuksia toteutettavaksi Wapice Remote Management (WRM) etähallintajärjestelmään. Erityisenä tutkimuskohteena on ISO 11783 standardin tarjoamat valmistajariippumattomat rajapinnat.

Työ jakaantuu kolmeen pääosaan. Ensimmäisessä osassa esitetään maatalousjärjestelmien nykytilanne, jonka jälkeen esitetään esimerkkisovelluksia, jotka myös määrittävät järjestelmän tärkeimmät vaatimukset. Toisessa osassa esitellään SAE J1939 -protokolla, johon ISOBUS perustuu. Tämän jälkeen käydään läpi ISOBUS-protokollan tärkeimpiä käsitteitä sekä verrataan ISOBUS-protokollaa muihin yleisiin CAN-pohjaisiin protokolliin. Tutkimustulokset ISOBUS-järjestelmissä saatavilla olevasta telematiikkatiedosta, sekä esimerkki järjestelmän yleisestä arkkitehtuurista on myös esitetty. Kolmannessa osassa kuvataan ISOBUS-tuen toteutus WRM-järjestelmässä sekä varsinaisten telematiikkatoiminnallisuuksien toteutus.

Työn tulokset näyttävät, että ISO 11783 -standardiin perustuva ISOBUS-protokolla tarjoaa monia toimintoja sekä rajapintoja, joiden avulla telematiikkatietoa voidaan kerätä valmistajariippumattomalla tavalla. Tämä on tärkeää, kun kehitetään yleiskäyttöistä järjestelmää. Lisäksi esimerkkitoteutus osoittaa, että ISOBUS-telematiikkaa voidaan integroida olemassa olevaan etähallintajärjestelmään.

PREFACE

This Master's thesis was written for Wapice Ltd. as a part of EFFIMA Agromassi research project. The thesis was supervised by Tommi Moisio from Wapice, and the examiner was professor Hannu Koivisto.

First and foremost, I would like to thank Wapice Ltd. for an interesting topic, guidance and also for the equipment needed to carry out the thesis. I would also like to show my appreciation for my co-workers at Wapice, and for the individuals I have had opportunity to work with in the Agromassi project. Special thanks to Tommi Moisio who supervised the thesis and also provided valuable comments and tips. I would also like to thank my examiner, professor Hannu Koivisto, for constructive comments and valuable feedback throughout the writing process. Finally, I would like to thank my family for their support throughout my studies.

Tampere, on May 3th 2014

Sami Pietikäinen

CONTENTS

TERMS AND ABBREVIATIONS

AEF	*Agricultural Industry Electronics Foundation*, an organization that develops and maintains the ISOBUS and its conformance testing.
Agromassi	*Assisting and Adaptive Agricultural Machine*, a research project that aims to research and develop self-adjusting, adaptive and inherently safe assisting control functions for agricultural machines.
CAN	*Controller Area Network*, communications bus which is used especially in automotive and industrial applications.
CAN Kingdom	CAN-based higher layer protocol that is maintained and developed by Kvaser.
CANOpen	CAN-based higher layer protocol that is maintained and developed by the CiA.
CiA	*CAN in Automation*, an international organization that develops and maintains high-level CAN-based protocols.
CIP	*Common Industrial Protocol*, higher layer protocol used in the automation industry.
DeviceNET	Higher layer automation industry protocol that uses CAN-bus and CIP-protocol.
CLAFIS	*Crop, Livestock and Forest Integrated System*, FP7 funded research project.
DMA	*Direct Memory Access*, a feature in computer systems which allows hardware to access main memories independently of the central processing unit.
DTC	*Diagnostics Trouble Code*, a numerical trouble indicator that is sent by an ISOBUS ECU.
ECU	*Electronic Control Unit*, physically independent electronic unit that may contain both electronic and software modules.
EFFIMA	*Energy and Life Cycle Cost Efficient Machines*, FIMECC-project which aims to develop new technologies to lower life-cycle costs.
FIMECC	*Finnish Metals and Engineering Competence Cluster*, consortium for companies and research organizations to help innovate and develop new technologies.

FMI	*Fault Mode Indicator*, Numerical code that identifies failure type in ISOBUS diagnostics messages.
FI-PPP	*Future Internet Public-Private Partnership*, European programme which aims to accelerate the development and adoption of Future Internet technologies in Europe.
FI-WARE	*Future Internet Ware*, project that aims to develop truly open, public and royalty-free architecture and core platform for future internet applications.
FMIS	*Farm Management Information System*, a software system to collect, process and analyse information to help manage operations of a farm.
FP7	*Seventh Framework Programme*, one of the financial tools through which the European Union supports research and development activities.
HAL	*Hardware Abstraction Layer*, software layer that contains hardware dependant parts to improve portability.
Headland	Strip of land along the edge of an arable field left unploughed to allow space for machines.
HTTP	*Hypertext Transfer Protocol*, request-response protocol for distributed, collaborative, hypermedia information systems.
Implement	Device that is mounted either rear or front of the tractor and is used for agricultural work.
ISO	*International Organization for Standardization*, an international standard-setting body composed of representatives from various national standards organizations.
ISOAgLib	A software stack that provides generic implementation for ISOBUS protocol.
ISOBUS	Communication protocol for agricultural industry which is defined in ISO 11783 standard.
ISOBUS stack	Software component that encapsulates ISOBUS-protocol specific implementation.
LinCAN	CAN-bus driver for Linux.

LLC	*Low Level CAN*, Wapice developed software component that provides higher level access to CAN-drivers.
NAME	64-bit globally unique identifier for a device in SAE J1939 -based networks.
NMEA	*National Marine Electronics Association*, organization for maritime industry in the United States.
NMEA 2000	Higher layer protocol that is based on SAE J1939 and is used in maritime applications.
Observer pattern	Software design pattern where observable object provides automatic notifications to its observers.
OSI	*Open System Interconnection*, a standard that describes a layered reference model for a telecommunication network.
PDU	*Protocol Data Unit*, a CAN-frame in SAE J1939 based networks, including ISOBUS.
PGN	*Parameter Group Number*, an identifier that specifies the message type in SAE J1939 based networks, including ISOBUS.
PLC	*Programmable Logic Controller*, a programmable digital computer system targeted for industrial control applications.
REST	*Representational State Transfer*, an architectural model to implement software interfaces based on HTTP-protocol.
RTOS	*Real-time Operating System*, operating system that is designed to guarantee deterministic performance which is required in real-time applications.
SAE	*Society of Automotive Engineers*, standardization organisation for automotive industry in the United States.
SAE J1939	CAN-based higher layer protocol that is maintained by SAE.
SDS	*Smart Distributed System*, a CAN-based higher layer protocol for industrial automation.
Singleton pattern	Software design pattern that restricts the instantiation of a class to one common object.

SOAP	*Simple Object Access Protocol*, a protocol for information exchange between Web services.
SocketCAN	CAN-bus driver for Linux.
SPN	*Suspect Parameter Number*, an unique identifier that identifies the contents of a data element in SAE J1939 -based networks, including ISOBUS.
SSH	*Secure Shell*, a cryptographic network protocol for secure communication that is used, for instance, to gain a remote command line access.
WCC-driver	*Wapice Custom CAN*, CAN-driver developed by Wapice Ltd.
Working-set	An integrated working unit that contains a tractor and one or more implements.
WRM	*Wapice Remote Management*, a complete remote management system developed by Wapice ltd. that contains electronis, servers and sofware.
XML	*Extensible Markup Language*, a markup language designed for structuring of information.

1. INTRODUCTION

Traditionally the agricultural equipment manufacturers have been mainly hardware providers, and the product life cycle services have been limited to spare parts and machine maintenance. The modern wireless technologies makes it now possible to collect data directly from the machine in its working environment. This data can be used to build a wide range of services covering the whole product life cycle. Information from multiple working units can also be combined and used for efficient fleet management or the information can be combined with other data sources such as weather forecasts to built sophisticated applications.

The main motivation to implement such services is to provide more value to the end customer. In practice this can be, for instance, enhanced servicing or better yield through monitoring and optimizations. By providing these services, the machine manufacturer can make itself more competitive in the global market. There are, however, conditions that need to be met before these functionalities can be implemented. The machine needs to be accessible remotely in its working environment and the data collected from it should be accessible to build the actual services.

To meet these conditions a telematics system is implemented which provides the infrastructure to built the actual value adding services on top of it. The telematics system provides connectivity to individual machines as well as aggregates data from multiple machine instances. It also provides well-defined interfaces to use its resources to built the actual services. However, a mere connectivity to the machine is not sufficient. The telematics system also needs to be able to communicate in the machine's internal communications network which in agricultural equipment is increasingly often ISOBUS.

On a high level the telematics system consists of two main subsystems: remote devices and server system. The remote devices are located in the actual machines, and they handle the low level ISOBUS communication in the machine's internal network. They also provide the server system with data collected from the ISOBUS network. The server system is responsible for storing the data and providing interfaces for outside world.

The purpose of this thesis is to study the available telematics information in an ISOBUS system, and also the applications this information can enable. Based on this information telematics functionalities are selected for implementation in the

Wapice Remote Management (WRM) system. The main goal is to have ISOBUS support in the WRM system, and also ISOBUS telematics functionalities that use standard manufacturer independent interfaces.

In whole, the telematics system is complex, so the main emphasis in this thesis is on the high-level architecture and interfaces. Therefore the detailed structure of the underlying components is not discussed in detail. The implementation and design of the value adding services built on top of the telematics system is also beyond the scope of this thesis.

Chapter 2 begins by presenting the current state-of-the-art in agriculture and agricultural research. It then goes through usage examples which highlight the different applications and requirements for the system in whole. This chapters gives a practical context for the actual ISOBUS telematics system in agricultural applications.

Chapter 3 begins by discussing the need for a communications network in agricultural machines in general. The Chapter 3 also introduces SAE J1939 protocol which forms the basis for ISOBUS. Thereafter, Chapter 4 gives an introduction to the ISOBUS protocol and the ISO standard it is based on. The most common device types in an ISOBUS network are also introduced. Followed by ISOBUS, Chapter 5 briefly introduces other CAN-based protocols. These three chapters give the necessary technical basis needed to understand the implementation of the telematics system.

Chapter 6 goes into the actual telematics system by presenting a high-level example reference architecture. This chapter also presents the results for the study of available information sources in an ISOBUS system. Thereafter, Chapter 7 introduces Wapice Ltd. as well as the Wapice Remote Management system in which the ISOBUS telematics functions are implemented. Chapter 8 then presents the implementation of ISOBUS support and telematics functionalities in WRM system. Finally, Chapter 9 summarizes the main topics and results.

This thesis was written as a part of the Agromassi project (Assisting and Adaptive Agricultural Machine) in which the author is participating on behalf of Wapice Ltd. Agromassi is conducted in co-operation with Aalto University, MTT Agrifood Research Finland, University of Helsinki and 11 agricultural machinery manufacturers and software vendors. The project is part of the Energy and Life Cycle Cost Efficient Machines (EFFIMA) research programme, which is managed by the Finnish Metals and Engineering Competence Cluster (FIMECC) and funded by the Finnish Funding Agency for Technology and Innovation (TEKES) together with participating research institutes and companies.

2. APPLICATIONS FOR TELEMATICS

There is a lot of information available in agricultural machines that would also be beneficial to have available outside the machine. Such information could be for example fuel consumption, vehicle speed and machine location to mention a few. Telematics system provides a means to expose interesting information from the machine for other, external, applications to use. It also makes it possible to gain remote access to the machine.

This chapter describes some of the applications in which the telematics information could be utilized. These applications also highlight the characteristics which are needed from the telematics system in whole. Some of the applications need an ability to communicate with the machine in real time, whereas others will need history data collected over a longer period of time.

Chapter 2.1 first introduces the extended product concept giving a larger context and a business view for the telematics system. Thereafter, Chapter 2.2 presents the current state-of-the-art in agriculture from the telematics point-of-view. Chapter 2.3 then presents usage examples in which the telematics information could be used. Finally, Chapter 2.4 addresses information security aspects.

2.1 Extended product

The customer focus has shifted from the ownership of a physical product towards an integrated solution that has been built around the core product. The integrated services not only help a manufacturer to compete in a global market, but can also provide additional revenue throughout the product life cycle. [1, p. 40]

Extended product means a core product that has been extended by offering additional value adding services to the customer. These services or assets are usually information and knowledge intensive, and can consist of engineering, software, maintenance, customer support and many others. [1, p. 40] Figure 2.1 shows a core product that has been packaged to a customer appealing tangible product. The tangible product has then been further extended by providing non-tangible, for instance digital, services.

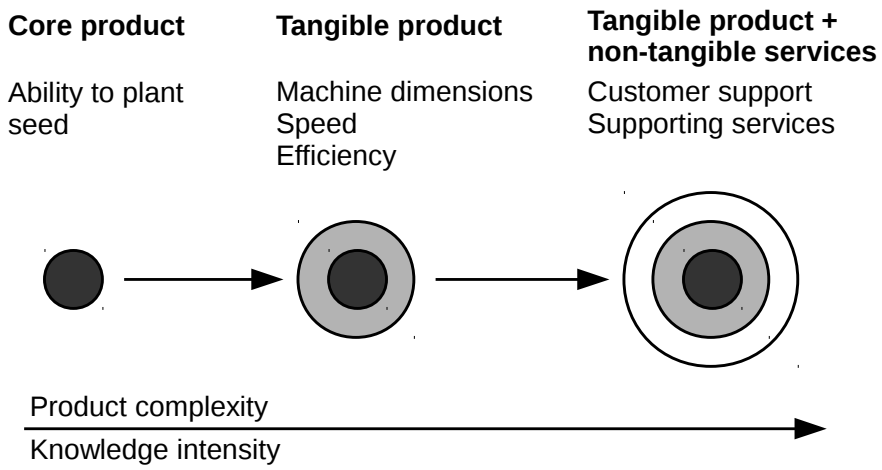

Figure 2.1: Extended product concept.

The telematics system can be used as an infrastructure to provide information and tools that allows manufacturers to create new concepts around the core product, thus extending their product. The telematics system can for instance gather data or provide a means to gain remote access to the machine. These assets can then be used to provide services directly to the customer or indirectly by, for example, enhanced servicing.

2.2 State-of-the-art in agriculture and agricultural research

ISOBUS provides technology to integrate the controls of tractors and implements to an integrated working-set which makes the driver's work easier and makes it possible to use machines with different tractors in a plug-and-play fashion. ISOBUS also provides technology for location-based precision farming by defining tasks which are planned in farm management information system (FMIS) and then executed by the ISOBUS working-set. The technical aspects of ISOBUS are discussed in the Chapter 4.

Based on discussions with Finnish agricultural equipment manufacturers during the Agromassi project, the current situation especially in Finland is that there are not many ISOBUS-enabled tractors or implements actually in use. This is partially caused by limited selection of ISOBUS equipment in the market. For instance, from Finnish implement manufacturers currently only Junkkari offers ISOBUS option to one of it's machines. Furthermore, the machines that do support ISOBUS usually only provide basic functionality such as user interface using the ISOBUS display (Universal Terminal). The customers in the agricultural domain also tend to be conservative about new emerging technologies.

Farm management information systems are software systems which collect, store,

process and analyse data to help manage the operations of a farm. These operations include for instance resource monitoring, work assignment planning, documentation and assessment of performed work assignments. In the article Farm management systems and the Future Internet era Kaloxylos et al. describe visions for future FMISs but also their current state. Usually the FMIS is a independent software running on the farmers PC, and all interaction with other systems such as weather forecasts or government systems are explicitly handled by the FMIS provider. Most of the features target simple monitoring, planning and bookkeeping, and advanced features such as decision support are not available. [2]

An extensive FMIS research was also carried out by Wageningen University within the Dutch Program on Precision Agriculture in 2012. The research comprised over 80 FMIS applications (different versions from same software were reviewed separately) from around the world. The main conclusion was that integrated solutions are very rare. Usually farmers need to use multiple different applications to support all of their farm management processes, and the data exchange between these applications is not automatic. Many of the applications are also very regionally focused. [3]

From the telematics point of view neither the current machine-base nor the FMISs allow to use ISOBUS telematics widely. However, when the ISOBUS equipment becomes more common, it also allows to create more sophisticated applications in addition to the core ISOBUS technologies. Still, it seems that ISOBUS will be the main communication standard in agricultural machines in future. This is indicated by the foundation of Competence Center ISOBUS, which is an association founded by large German agricultural manufacturers in 2009 to promote ISOBUS development [4]. Similar association has also been founded in Finland in 2014 (Suomen Maatalousautomaatio ry). Furthermore, when looking at already certified ISOBUS products from one of the test laboratories DLG e.V., large international manufacturers such as John Deere and AGCO Group are present [5]. Therefore the research and development for ISOBUS telematics at present can provide valuable advance in the near future.

ISOBUS related research has been carried out in Finland for more than ten years. Between the years 2003 and 2005 Agrix project, that was led by Helsinki University of Technology (now Aalto University), researched core technologies in agricultural automation and precision farming, including ISOBUS. The project goal was to develop a prototype of an open, generic and configurable automation platform for agricultural machinery. Other partners included MTT Agrifood Research Finland, University of Helsinki and industrial partners from agricultural domain such as Junkkari, Valtra, Tume-Agri and others. [6]

Agrix project was followed by Farmix project which started in 2006 and ended in 2008. The Farmix project used information and experience obtained in the Agrix

project as background to further research the automation in agricultural machines. The main goal was to clarify challenges in tractor-implement integration and develop methods to integrate the control over tractor and implements. When the tractor-implement system is an integrated working-set, the driver's work becomes easier and also quality, efficiency and safety are improved. Many of the partners that participated Agrix were also involved with Farmix. [7]

After Farmix the research has been continued in the Agromassi project. At this point also Wapice got involved with the agricultural research. Agromassi started in 2009 and is due to end in 2014. In Agromassi project self-adjusting, adaptive, inherently safe assisting control functions for agricultural machine are developed to support future product development in companies. [8] One of the work packages targets telematics in agriculture in which this thesis is related to.

Agricultural research that relates to telematics has also been done in the Europe. For instance, FutureFarm project that was funded by the European Commission Seventh Research Framework Programme (FP7) was started in 2008. The project had partners from ten European countries, and the main goal was to research how to meet the challenges of the farm of tomorrow by integrating FMIS to support real-time management decisions and compliance to standards. The project duration was 3 years. [9]

Another FP7 funded research project that targeted agriculture was SmartAgriFood which started in 2011. SmartAgriFood was part of Future Internet Public-Private Partnership (FI-PPP), and the project goal was to research and promote the use of future internet technologies in agriculture. The project duration was two years. The interfaces and specifications done in the project were also aligned with non-agriculture specific Future Internet Ware project (FI-WARE). [10]

One of the most recent research projects that also targets agriculture is Crop, Livestock and Forest Integrated System (CLAFIS) which started in December 2013. The project aims to bring together technologies, experience and research results from industrial automation, Internet of Things and agribusiness sectors to address the needs for seamless data transfer between field devices, automation systems and IT systems for various stakeholders. [11] These topics are very closely related to ISOBUS telematics when the mentioned automation system is tractor-implement working-set. Wapice and the author are also participating CLAFIS which means that the results and experience obtained from this thesis may directly be used as project background.

The research topics in FutureFarm and SmartAgriFood targeted high-level systems such as FMIS and decision support systems, and the use of internet technologies in their realization. Agrix and Farmix on the other hand focused mainly on the the core ISOBUS technologies and precision farming in tractor-implement working sets.

In this thesis the potential and limitations of connecting ISOBUS working-sets to high-level systems similar to the ones researched in FutureFarm and SmartAgriFood in a manufacturer independent way is researched. This includes identifying different usage scenarios as well as ISO 11783 standard study to identify the different data sources and interfaces that are available in an ISOBUS network.

2.3 Usage examples

This chapter describes some of the applications in which the telematics could be utilized. The different cases highlight the various requirements for the telematics system ranging from the real-time remote access to data storage. These usage examples also show the different actors. Some of the applications are used by the end-customers, whereas others are used by the machine manufacturer.

The examples are discussed in the concept level, and their detailed technical implementation is not addressed in this thesis. The main purpose is to present a larger context for the telematics system. The applications are also not limited to the ones described in this chapter, and it is possible to come up with far more uses.

2.3.1 Remote diagnostics

Diagnostics are usually needed when a machine is not functioning as it should, or there is a problem with the machine installation. Typically these kind of situations occur on the field when the machine is in use. Without an access to the machine, it can be really difficult to diagnose the origin of the problem.

The problems can either be related to physical devices or software. Physical issues can be caused by, for instance, incorrect hydraulic or electrical connection or broken sensors or actuators. Software issues can be related to the device itself, or they can be incompatibility issues when operating on a shared bus.

Traditionally there have been two main options to diagnose the machine. The operator contacts the manufacturer, and the problem is diagnosed remotely, for example, by phone. Usually the operator does not have special diagnostics devices or expertise so the amount of information available for the support personnel is limited. Second option is to diagnose the problem on site by service personnel which can be really costly, and also means longer down-time for the machine. Currently most of the agricultural machines are not able to connect to the Internet which means that online diagnostics are not available, and the options described above are used. Moreover, the diagnostics are especially challenging when machines are sold abroad to countries which may not have the manufacturer's own service organization available.

As a part of the SmartAgriFood project a research of farmers needs was carried out. This research included both farmer interviews as well as scientific research of farm management systems that had been developed in earlier research projects such as FutureFarm, agriXchange, PPL, iGreen and others. Among these use cases were "Remote machine diagnostic" and "Faulty operation of sensors inside a farm". [2] This shows that the need for remote diagnostics has also been identified in other research projects.

The telematics system can be used to make diagnostics information available for the support personnel when diagnosing the machine remotely. The telematics data could include for example bus status, individual sensor statuses and readings and active error codes. This information is useful when diagnosing both software and hardware related issues. To fully benefit from the telematics system, the support personnel should be able to communicate with the machine in real time. History data can also be useful if the problem occurred during a work assignment. Telematics device could also be retrofitted to old machines.

This idea could also be taken even further. The diagnostics information could be recorded to an electronic service record with other service actions such as maintenances or new parts. This record would show the service history for the whole machine life-cycle.

2.3.2 Firmware updates

As the amount of new functionalities and automation increases in machines, usually so does the amount of software components in them. Like the mechanical parts, the software also needs maintenance. The software maintenance can include for instance bug fixes and introduction of new functionalities.

Updating software on an agricultural machine is challenging because usually the machines are spread out geographically, and also not connected to the Internet. The lack of connectivity to the Internet means that traditionally the firmware updates have to be done on site. This is also the situation with most of the current agricultural machines. The telematics system requires remote connectivity, and this same infrastructure could also be used to download software updates to the machine. The online firmware updating was also among the use cases identified in the SmartAgriFood project [2].

In addition to the maintenance updates, it would also be possible to offer upgrade packages to customers. The upgrades could for example introduce completely new features or unlock or extend existing functionalities. This would be very cost-effective for the product manufacturer because there would be no need to send a mechanic on site or bring the machine to service.

2.3.3 Performance monitoring

When a machine is carrying out a work assignment, it is usually relatively hard to obtain detailed information about the assignment for later analysis. Current values can be displayed to the operator, and data can even be logged locally to the machine. However, the data still needs to be manually collected from individual machines before it can be analysed. Additionally, this method does not allow real-time remote monitoring.

Sørensen et al. report that "the farmer voice a need for additional information and advanced technologies to manage monitoring and data acquisition on-line in the field". The current field operations monitoring is very laborious because the information needs to be imported and transferred manually, and the different applications do not coordinate with each other automatically. [12]

With the remote telematics, the machine data can be collected and stored to a server during the work assignment. It then becomes available for other applications which can combine, process, analyse or otherwise refine the data. Data from the whole machine fleet can also be combined for more detailed analysis.

It is also possible to equip the machine with positioning capabilities. This means that the machine data can be coupled with location information which makes spatial analysis possible. Location information could also be used for fleet management as the machines could be monitored in a near real-time.

The performance monitoring could be an individual system but it could also be integrated to a farm management system. In this case the performance information would be directly available, for example, to optimize future work assignments.

2.3.4 Integration to farm management systems

ISOBUS also allows to perform precision farming. The FMIS is used to plan work assignments, tasks, which are then performed in the machine. The tasks define how much resources, such as fertilizer or seed, are used in different parts of the field. After planning, the ISOBUS formatted tasks are transferred to the machine for execution, and after the task completion results are transferred back to FMIS. The device that orchestrates the task execution in the machine is called a task controller. [14] Usually the task transfers has to be done manually using for instance an USB drive. The telematics infrastructure could be utilized to transfer the task to machine and results back to the FMIS to provide an integrated solution for the farmer.

The integration to the farm management information systems does not need to be limited to transferring ISOBUS tasks. Other additional information could also be transferred to support the decision making and planning. For instance fuel consumption could be utilized for cost calculations or task completion times could

be used to schedule multiple tasks. Figure 2.2 shows how intelligent decision support system can be used to improve decision making. It is worth noting that the ISOBUS task controllers do already record the process variables during the work assignment, and the data can be coupled with the location information. However, the decision support system in the following figure encompasses a much more comprehensive knowledge than a record from a single work assignment.

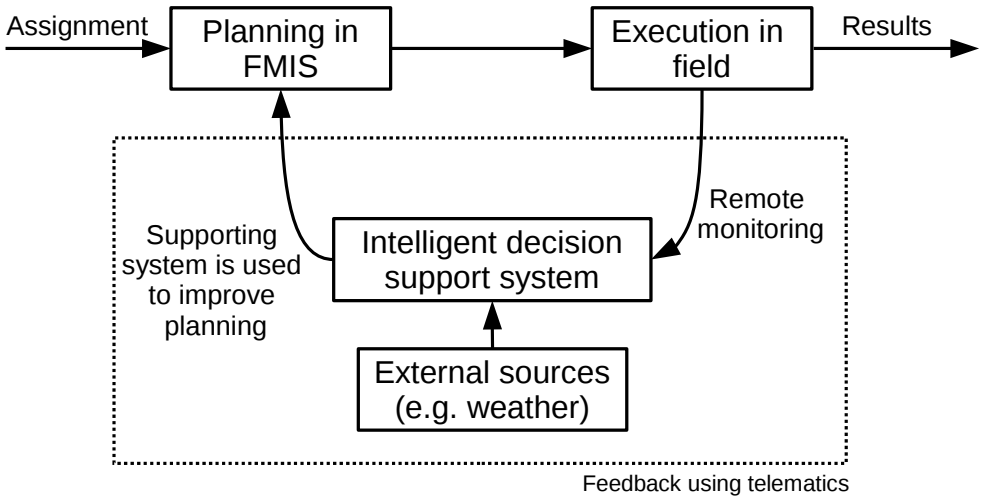

Figure 2.2: Work planning using intelligent decision support system.

When a new work assignment is received, it is planned in the farm management information system and then executed in the field. During the work assignment a wide range of parameters can be monitored and stored to a database. When a similar work assignment is planned in the future the experience from the previous work can be utilized to improve the decision making process. Currently the FMIS systems do not provide tools for intelligent decision making, and many farmers base their decision on implicit knowledge, intuition and formed routines [2].

More complex information can also be derived from the collected data. For example, a correlation between the amount of fertilizer used and the yield at the end of the growing season. In addition to the data collected from an individual farm, other data sources such as weather forecasts, anonymous data from other farmers or information from government systems could be used to improve the decision making.

In addition to the improved work planning, the telematics system integrated FMIS could also automate farm bookkeeping such as resource consumptions in different fields. It could also assist the farm manager with statutory forms and reports or give warnings if the planned work assignment does not meet requirements or recommendations.

2.4 Information security

An important aspect that needs to be addressed when designing the telematics system and also applications on top of it is information security. The information security should be taken into account in the practical implementation by using appropriate encryptions for the data and communications when it is sensible. There are, however, also other information security aspects that are not as apparent.

The collected information can give a detailed insight how a farm is being operated, and the end-user does not want this information visible to competitors. However, the telematics system does have many actors that are interested in different types of information. Therefore a proper access right management is needed. There is also a question of trust between the system provider and farmers.

Another interesting aspect, that is not strictly information security matter, is the question of ownership. There are many actors as well as groups of interest that are using the information from the telematics system directly or indirectly, but the ownership of the data is not so apparent. When the data is stored to the telematics system databases, is it owned by the telematics system provider, end-user or perhaps the machine manufacturer? This question should be answered when the telematics system and its applications are productised.

The data ownership, information security and trust were researched in the Smart-AgriFood project. These results are publicly available in the project deliverable D100.1 Review of the Literature and Future Internet Research [13]. The information security aspects are not discussed further in this thesis but they are nevertheless a very important part of the system.

3. DEVELOPMENT TOWARDS ISOBUS

When devices become more and more complex their control systems also need to evolve to meet these challenges. The early agricultural machines were first manually operated, and later electrical and hydraulic technologies provided means to implement more complex features. The modern digital systems have provided technologies to implement very sophisticated automation and control to assist the driver and to improve the efficiency of work.

This chapter describes the need for a communications bus, such as ISOBUS, in an agricultural environment. Also, another communications network, namely SAE J1939, is introduced in this chapter because it provides the basis for ISOBUS protocol.

3.1 Overview

In an agricultural environment it is common to have one or more tractors and also a number of implements that can be connected to these tractors. There are also many different manufacturers that provide both tractors and implements, and therefore the number of different combinations is very large. The figure 3.1 shows a typical tractor-implement configuration which consists of a tractor and rear and front implements.

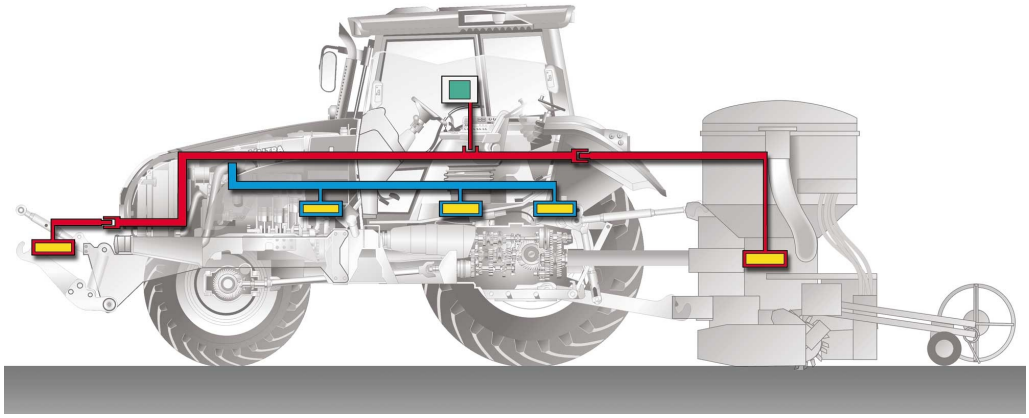

Figure 3.1: Tractor and implements [15].

Traditionally, if the implement needed some special controls besides the tractor hydraulics, they needed to be custom fitted to the tractor cabin. This, of course, makes the implement installation laborious, and if many different implements are used with the same tractor, the number of individual controls in the cabin can also became an issue. [16, p. 10] Additionally, it is also hard to swap the implement to a different tractor as it would require a re-installation of the controls. Figure 3.2 shows an example of dedicated control panel for Junkkari implements.

Figure 3.2: Junkkari Wizard Plus dedicated control panel [17].

These challenges were noticed by the agricultural industry, and the solution was to develop a implement bus where implements, tractor and other devices such as displays could communicate with each other in a standardized way, thus making it easy to install new devices to pre-existing systems. The purpose of the ISOBUS is just that. [16, p. 10] The ISOBUS was not, however, built entirely from ground up. Instead, SAE J1939 standard that was already in use in the automotive industry was used as a basis for the ISOBUS standard.

3.2 SAE J1939

SAE J1939 is a network protocol that was designed for in-vehicle communication in trucks and buses. The standard was first published in the late nineteens and is still developed by the Society of Automotive Engineers (SAE). The SAE J1939 is in turn built upon Controller Area Network (CAN). [18]

CAN is a network protocol which was originally developed for the automotive industry in the 1980s, and it is still widely used today. The CAN protocol has been standardized in the international ISO 11898 standard, and it defines the data link layer of the ISO/OSI reference model. [19] It is possible to use CAN as is, but usually a higher layer protocol is used to define how the CAN identifier and data bytes should be interpreted which makes, for instance, interoperability between manufacturers easier. SAE J1939 is a such higher layer protocol.

In SAE J1939 network only extended CAN-frame format with 29-bit identifier field is used for communication. Standard CAN-frames can also exist in the network but they are explicitly reserved for proprietary use. The CAN-identifier (CAN-ID) in SAE J1939 network is divided into Parameter Group Number (PGN) and source address which identifies the node that sent the message. In turn, the PGN is also divided to multiple fields which are shown in the figure 3.3. [18]

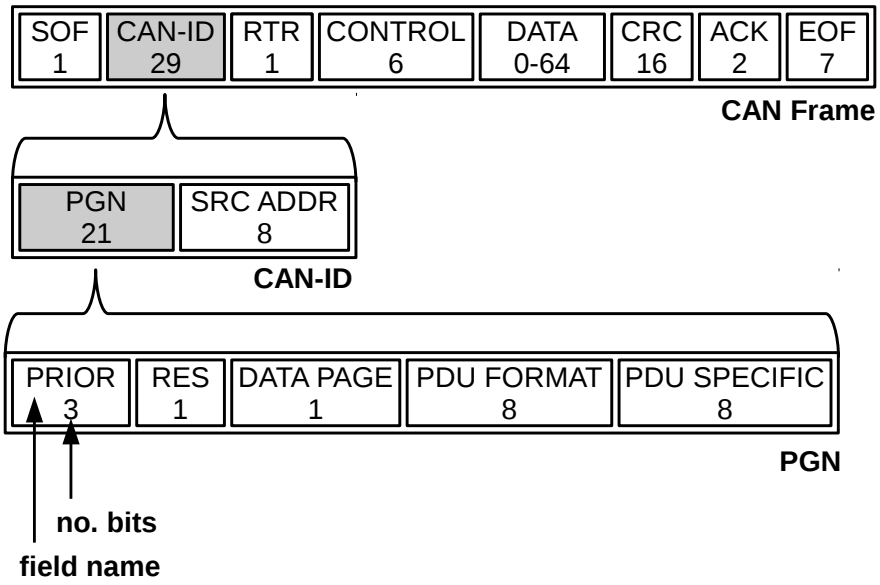

Figure 3.3: CAN frame structure in SAE J1939.

The priority (PRIOR) field is used for bus arbitration, priority 0_{10} being the highest priority. The bit after priority is reserved for future extensions. The next field is data page which can be used in future to expand the amount of PGNs. The Protocol Data Unit (PDU) format field specifies, how the PDU specific field should be interpreted. When the PDU format is between 0_{10} and 239_{10}, the PDU specific field is interpreted as destination address. Messages in this PDU format range are intended to be used for peer-to-peer communication between two nodes. When the PDU format is above 239_{10}, the PDU specific field is interpreted as group extension. This message range is used for broadcast messages, and the group extension makes it possible the express more unique message types. [18]

When using peer-to-peer communication, each node in the network needs to be explicitly identified so that a message can be directed to it. In SAE J1939 each device is identified by a globally unique 64-bit NAME which specifies, for instance, the device type and manufacturer. The NAME is the machine's persistent identity which does not change runtime. The structure of the NAME is shown in the table 3.1. [20]

Table 3.1: SAE J1939 device NAME.

Field	**Bits**
Arbitrary address cabability	1
Industry group	3
Vehicle system instance	4
Vehicle system	7
Reserved	1
Function	8
Function instance	5
ECU instance	3
Manufacturer code	11
Identity number	21

The NAME cannot be used as the message address as is because the CAN-ID is only 29-bits long, and the whole data field in CAN frame is only 64-bits. Instead, a shorter one byte address is used in communication. Valid addresses for devices are 0_{10}-253_{10}, 254_{10} is null address and 255_{10} is broadcast address [21, p. 5]. In SAE J1939 network there is no master node that could hand out these addresses for devices. Instead, the bus addresses are obtained using an address claim procedure specified by the standard. When a new node joins the network, it first needs to obtain a valid address to start communicating. When claiming an address, the NAME is placed in the CAN data field as shown in the figure 3.4 to identify which device wants to claim an address. [21, p. 5]

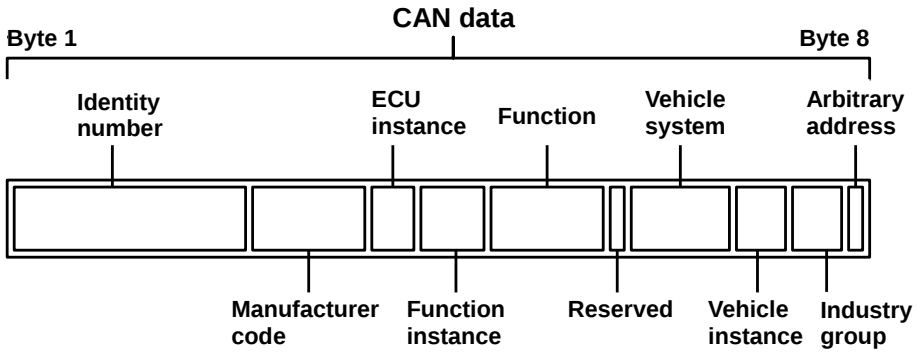

Figure 3.4: Structure of the CAN data field when NAME is sent.

Figure 3.5 shows a full address claim procedure without any conflicts. First, the device joining the bus sends a request for address claim (1.). The new device does not yet have a valid address but it is allowed to use the null address 254_{10} as a source address. Destination address is set to broadcast (255_{10}). When a request for

address claim is sent to the network, all other nodes must reply with address claim message using their own source address (2. and 3.). The address claim message's data field contains the 64-bit NAME to associate the device with the source address. Now the new device can detect all the addresses that are already in use, and it can select a free address. The device then claims the address by sending an address claim message using the newly selected address as the source address (4.). This way the other nodes can know which address the device is going to use. Finally, the device needs to wait if there is address conflict before it can start the normal communication using the claimed address (5.). [21, p. 6-7]

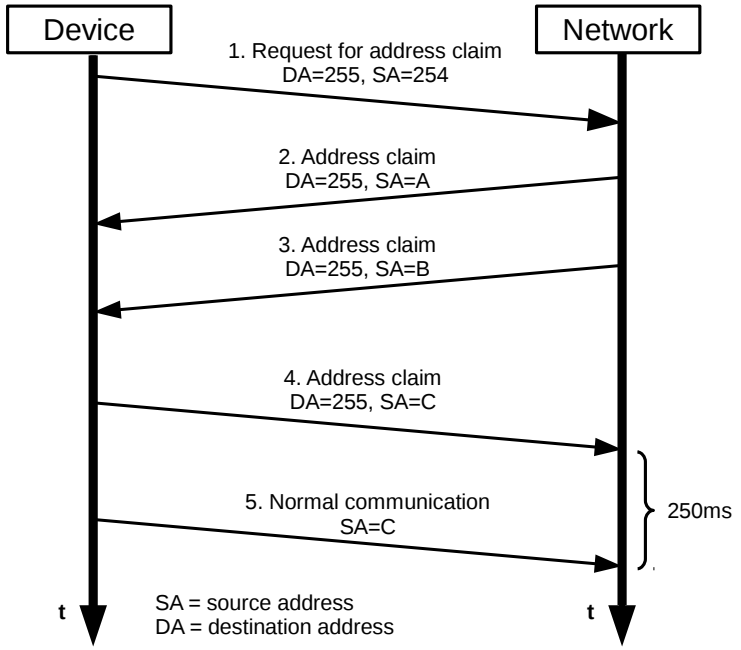

Figure 3.5: Address claim procedure without conflicts.

In addition to the network management functionality, the SAE J1939 also defines application level messages. This is done by assigning meanings to the individual protocol group numbers. For instance, the PGN 65257_{10} is "Fuel Consumption (Liquid)" message [22]. Each message can also contain multiple data fields which are also defined by the SAE J1939 and identified by their Suspect Parameter Number (SPN). The "Fuel Consumption (Liquid)" message, for instance, contains SPNs 182_{10} and 250_{10} which are "Trip Fuel" and "Total Fuel Used" respectively [22]. The standard also defines length, resolution and offset for each suspect parameter number.

Basically the SAE J1939 is a proven protocol for in-vehicle communications, and it is also largely adopted by, for instance, diesel engine manufacturers. The protocol defines both network management functionality as well as application messages. Therefore the standard series provided a solid basis for ISOBUS.

4. ISOBUS

ISOBUS is an universal protocol for communication between implements, tractors and computers in an agricultural environment. The primary goal for ISOBUS is to ensure compatibility between machines from different manufacturers, and also between the mobile systems and the management software used on the farm. ISOBUS is managed and developed by Agricultural Industry Electronics Foundation (AEF). The basis for ISOBUS is the ISO 11783 standard - "Tractors and machinery for agriculture and forestry - Serial control and communications data network". [23]

This chapter introduces the main concepts of the ISOBUS protocol as well as the ISO 11783 standard in which the protocol is defined. The standard in whole consists of 14 parts and more than a thousand pages so all of the standard parts are not covered. Also, some of the most common device types in an ISOBUS network are introduced.

4.1 Overview

ISOBUS is based on the SAE J1939 protocol which was introduced earlier in the chapter 3.2. SAE J1939 was designed to be used in automotive applications which shares many of the requirements with the agricultural applications, or more generally moving machines. Therefore the J1939 was a suitable basis for the ISOBUS protocol. It also meant that there was no need to reinvent functionalities such as network management or device addressing because they were already proven and tested by the J1939. This also meant that the ISOBUS developers could only specify the agricultural specific messaging, and use the J1939 base mechanisms such as address claiming as is. The concepts described in chapter 3.2 also apply to ISOBUS and therefore they are not discussed again in this chapter.

As already mentioned in the chapter 3.1, a typical arrangement in an agricultural machinery usually consists of tractor, a number of implements and other devices such as displays. Figure 4.1 shows an example how these devices could be connected using the ISOBUS.

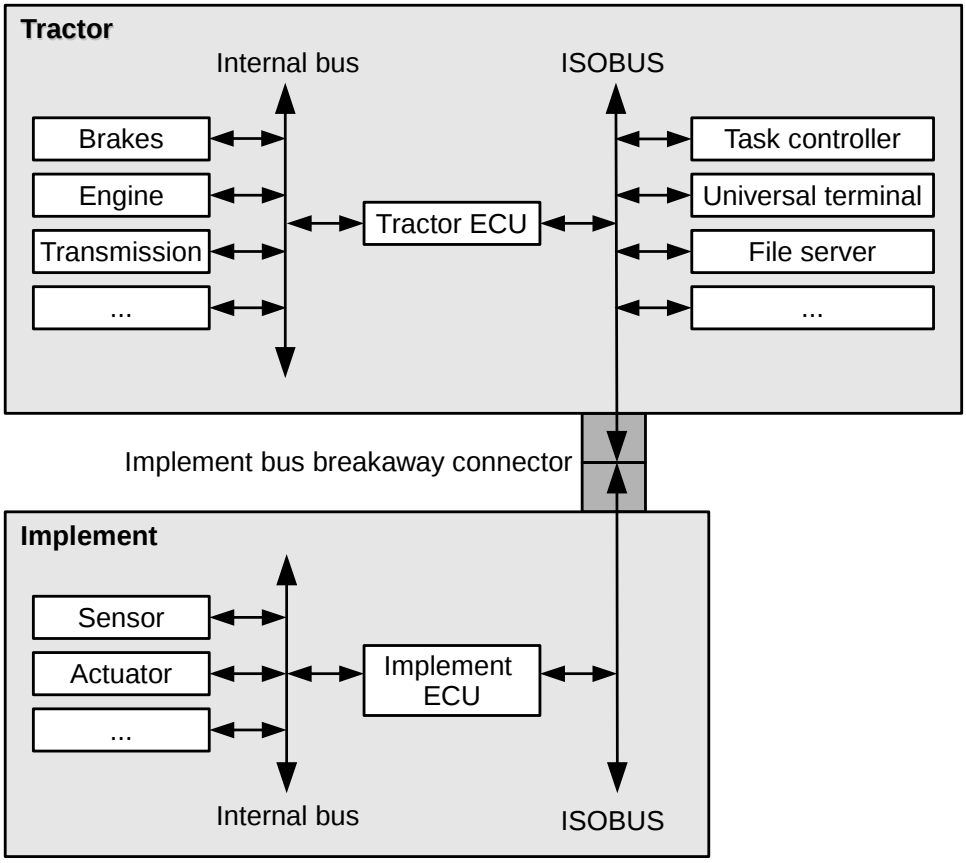

Figure 4.1: Device interconnections using ISOBUS. [24, p. 12]

The figure shows that the tractor has its own internal control system which is used for instance for motor and transmission control. This internal control system is not directly visible to ISOBUS devices. Instead, the tractor has a dedicated control unit which exposes some of the internal information to other ISOBUS devices. Similarly, the implement may have its internal connections to sensors and actuators, and the ECU (Electronic control unit) is the device that connects to the ISOBUS. The figure also shows other ISOBUS devices (Task controller, Universal terminal and File server) which are introduced later in the chapter 4.4. Both the physical connections and the communication between these, and also other ISOBUS devices, are specified in the ISO 11783 standard series.

The ISO 11783 standard currently consists of 14 parts [25] which describe the communications network for agricultural equipment from the physical level all the way to the application level. Part 1 is a general introduction to the standard series, and it also describes the most important concepts of the network. Part 2 defines the physical layer and connectors so that devices can be physically connected in a standardized manner. Parts 3 to 5 define the actual data link and network mechanisms which are similar to SAE J1939. Rest of the standard parts define application level

messages and mechanisms for agricultural applications. All of the standard parts are listed in the table 4.1.

Table 4.1: ISO 11783 standard parts [25].

Part	Name
1	General standard for mobile data communication
2	Physical layer
3	Data link layer
4	Network layer
5	Network management
6	Virtual terminal
7	Implement messages application layer
8	Power train messages
9	Tractor ECU
10	Task controller and management information system data interchange
11	Mobile data element dictionary
12	Diagnostics services
13	File server
14	Sequence control

When looking at the individual standard part names, many of them use a term layer. For instance part two is "Physical layer" and part three is "Data link layer". These layers refer to OSI reference model which is introduced in the following chapter.

4.2 OSI model and ISOBUS

Open Systems Interconnection or OSI in short is an architectural model which defines a partitioning of network functionality into seven layers, where one or more protocols implement the functionality in each layer. The model does not specify an individual protocol, but instead provides a reference model for the main functionalities needed for networking. [26, p. 27] The OSI model is specified in the ISO/IEC 7498-1 standard. The OSI model is shown in the figure 4.2.

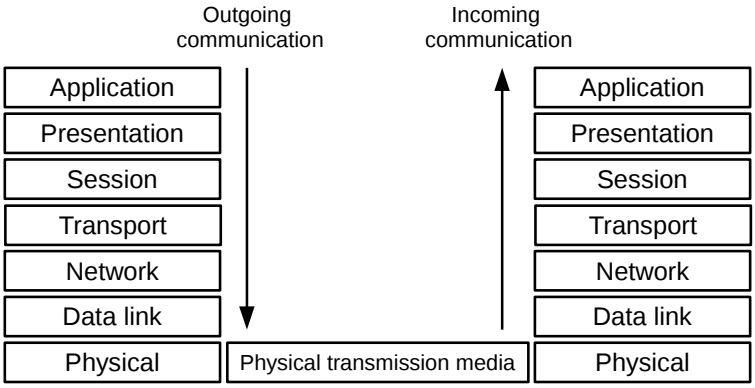

Figure 4.2: Open Systems Interconnection reference model.

It is not required that a standard or protocol based on the OSI model should be explicitly partitioned into the seven OSI-layers. It is also not required that the protocol should always provide all the layers if they are not needed for the specific application. This is also the case with ISO 11783 as it is designed to support a definite set of applications for a specific industry. The ISO 11783 standard specifies the application layer, and the four lowest layers of the OSI model which are physical, data link, network and transport layers. [24, p. 11]

The ISO 11783 does not specify session or presentation layers because they are not needed. Figure 4.3 shows the respective standard parts which specify each of the layers used in the ISO 11783 network. The network management has been represented in the left side of the OSI layers as a separate functionality. Basically the network management functionality consist of the automatic address allocation and management scheme which does not fit to a any single layer.

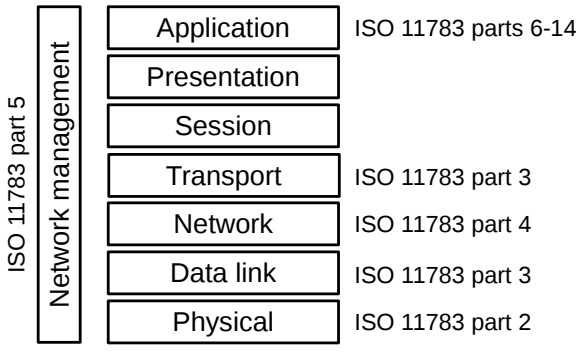

Figure 4.3: Application of OSI in ISO 11783.

The physical layer in ISO 11783 network is composed of a single linear quad-twisted wire cable which connects the devices. Also, active terminating bias circuits

are used in the each end of the network segment. The standard also specifies the connectors used to connect implements to tractor, additional ECUs to network segment and a service tool to the network. In addition to the wiring and connectors, the standard also specifies power sources needed by the network. [24, p. 11] So even though the ISO 11783 is based on SAE J1939, it still specifies additional requirements for the physical layer to ensure interoperability between devices.

The data link layer in ISO 11783 is based on the CAN extended frame format, and it is specified in the ISO 11783-3. CAN base format message frames can also be used in ISO 11783 network, but they are reserved exclusively for proprietary use. The data link layer in ISO 11783 specifies, how the extended CAN frame identifier should be interpreted. It also specifies two different message formats, one for broadcast-only messages and one for destination specific messaging. [27] In practise, this means the protocol group number mechanism that was already introduced with the SAE J1939.

It is also possible that there is a need to connect a network with a different architecture to an ISO 11783 network. For example machine's internal communication bus. ISO 11783 network layer specifies network interconnect units which shall be used to isolate network segments from each other. It is also possible to use network interconnect units between ISO 11783 network segments to accommodate more devices than electrical limits allow for a single segment. [24, p. 13]

The ISO 11783 part 3 which defines the data link layer also specifies a protocol for multi-packet messaging. [27] This functionality corresponds to the transport layer in the OSI model. This protocol is needed when more data needs to be transferred that can be accommodated to a single CAN frame.

ISO 11783 standard also specifies a lot of functionalities for the application layer. The different functionalities are grouped to individual standard parts. For instance, virtual terminal messages are defined in ISO 11783-6 and messages between implement and tractor are defined in ISO 11783-7. These standard parts describe how the lower layers are utilized for each individual functionality, and also how the data is represented in the message frame.

4.3 Communication in ISOBUS network

In an ISOBUS system the communication between devices is very tightly coupled to the protocol group numbers defined by the ISO 11783 standard. In practise, this means that the ISO 11783 standard defines the meaning for each PGN which in turn defines the message CAN ID. These PGN definitions are not limited to network management since the standard also defines the PGNs that are available for application level functionality. This is a similar practise that is used in SAE J1939.

This approach means that an individual ECU designer cannot extend or modify the functionality provided by the standard. When new features are introduced, new PGN values are assigned for them, and the functionalities are added to the standard series. These standard parts not only define the message PGNs but also the contents of the CAN data bytes. This methodology does not offer much flexibility for a designer but in turn it makes seamless interoperability between manufacturers possible. The table 4.2 shows a practical example of such application level message definition.

Table 4.2: Ground-based speed and distance message. [28]

Field	**Value**
Transmission repetition rate	100 ms
Data length	8 bytes
Data page	0
PDU format	254
PDU specific	73
Priority	3
Protocol group number	65097 ($00FE49_{16}$)
Bytes 1,2	Ground-based machine speed
Bytes 3 to 6	Ground-based machine distance
Byte 7	Reserved
Byte 8, bits 8 to 3	Reserved
Byte 8, bits 2,1	Ground-based machine direction

The "Ground-based speed and distance" message is usually sent by the tractor ECU in a constant 100 ms interval. From the message definition we can see that the PDU format is 254. This means that the message is broadcast and the PDU specific field is group extension. The standard also defines the meaning for each of the data elements. Similarly to SAE J1939, each data element has a SPN assigned for it. As an example table 4.3 shows how the "Ground-based machine speed" field is interpreted.

Table 4.3: Ground-based machine speed field. [28]

Field	**Value**
Data length	2 bytes
Resolution	0,001 m/s/bit, 0 m/s offset
Data range	0 m/s to 64,255 m/s
Type	Measured
Suspect parameter number (SPN)	1859

So basically large portion of the ISO 11783 standard series contains message and parameter definitions for various application level functionalities. Clearly, the mere message and parameter definitions are not enough to define complex functionalities. Therefore the respective standard parts also define the sequences how these messages are used to perform various functions such as updating the user interface.

The ISO 11783 also allocates PGNs for proprietary use. Messages that use these PGNs can be both broadcast or peer-to-peer, and both single- and multi-message communication are also allowed. These messages make it possible to implement functionalities that are not defined by the standard. However, the use of these functionalities are limited to devices that can understand this manufacturer specific messaging.

4.4 ISOBUS equipment

ISOBUS systems can contain a wide range of different types of devices from different manufacturers. This section briefly introduces some of the most important device types which are common for ISOBUS system. These are logical functionalities, and a same physical device may implement multiple of these types.

4.4.1 Universal terminal

Universal terminal is a ISOBUS specific display unit which is used to provide user interface for the machine operator [30]. Devices can load their user interface to the universal terminal, update values and receive user events from the terminal. This way there is no need to install device specific control panels because the user interface can be provided in the standardized way in the universal terminal.

Requirements for the universal terminal are defined in the ISO 11783 standard part 6. Tractor ECU and also other ECUs connected to the implement bus can utilize the universal terminal in a similar fashion as the implements. [24, p. 15] It is also worth noticing that nowadays the official term used by the AEF is Universal Terminal whereas the ISO standard uses the term Virtual Terminal. These both terms mean the same device.

4.4.2 Tractor ECU

Tractor ECU (TECU) is a network interconnect unit that connects the tractor's internal control network to the implement bus. It also provides electrical isolation between these buses. This isolation is necessary because the tractor's internal bus is used for safety critical communication such as motor and transmission control. The tractor ECU also works as a bridge if the tractor's internal bus uses a different communications protocol. [24, p. 15]

Tractor ECU can also expose information about the tractor's state to the implement bus by broadcasting the information. For instance wheel-based speed, power take-off rotation speed, hitch positions and much more can be sent. This way all other nodes that require this information can easily gain access to it. The different message types are defined in ISO 11783 standard part 7. [28] It is also worth mentioning that there are differences between tractors, and usually only a subset of the tractor messages defined in the part 7 are actually implemented.

Additionally to sending information to the implement bus, the tractor ECU can also receive requests from the bus. These request messages are also defined in the part 7 of the standard. This way the tractor hydraulics and the tractor speed and steering can be controlled. This makes it possible to implement automatic features to assist the driver but it also introduces functional safety issues.

4.4.3 Implement ECU

Implements are the actual machines that are connected to front or rear of the tractor. Different implements are needed for different work assignments so it is normal that the implements are changed frequently. As an example a combi drill or a sprayer would be considered as an implement.

In an ISOBUS system the implement provides its user interface using the universal terminal. It is also possible that multiple implements are using the same universal terminal simultaneously. The implement ECU can also support other interfaces in addition to the universal terminal. A common interface is task controller interface which can be used for precision farming.

4.4.4 Task controller

Task controllers are devices that provide scheduled control of implement functions via the ISOBUS network. The task controllers receive planned work assignments, tasks, from the farm management computer system. During the actual work, the task controller sends control messages to the implement network as planned in the task. The task controller also records data sent by the implement which can later be reviewed in the farm management system. [24, p. 15] It is also possible to combine positioning to the task which enables spatial precision farming.

The operation of the task controller and the messages sent to and from the implements are defined in the ISO 11783 part 10. The data elements that can be controlled by the task controller are defined in the standard part 11. The use of task controller requires that the implement ECU implements the task controller interface. This also includes a device description which lists data elements that can be controlled in the ECU in question. Different data elements can also be viewed

from the online ISOBUS Data Dictionary [29].

4.4.5 Other equipment

There are also other devices that can be used in the ISOBUS system besides the ones that have already been introduced in this chapter. One device that is needed for precision farming is a positioning system. The position information can be provided in the same format as in NMEA 2000 by the tractor ECU or by a separate positioning unit. The ISO 11783 standard also defines diagnostics devices that can be connected to the bus. These devices can be used to diagnose problems or configure the devices. Both proprietary and ISOBUS communication can be used by the diagnostics devices.

The standard part 13 defines file server functionality. File server devices can be used as a common storage devices. Other devices can both read and write files to these servers, and the standard also defines protected manufacturer specific storage which is only visible to the devices from the same manufacturer. [31] For instance an universal terminal could expose an USB stick to the bus using the file server interface, and reports or logs could easily be transferred to it.

The standard part 14 defines a device type which is relatively new in the ISOBUS world, namely sequence controller. Similarly to the task controller the sequence controller can also issue control requests to the implements and tractor. The sequence controller makes it possible to record and then later reproduce a sequence of actions involving the tractor and/or one or more implements [32]. This makes it possible for the operator to perform a complex sequence of operations automatically with one action. This is especially useful if the complex operation needs to be repeated often. For instance, lowering or lifting the machines in the headland.

The Universal terminal provides its user interface which touch-screen, physical buttons and dials or with a combination of the above. However, some implements need precision control for which these controls are not best suited. The ISOBUS also allows the operator to use auxiliary controls such as joysticks if the ECU supports them [30].

4.5 Conformance testing

ISOBUS devices are developed by a large group of manufacturers around the world. This means that the amount of different machine combinations that can exist is considerable. However, from the end-user point of view all of these combinations should work in a plug-and-play fashion. This is the reason why the AEF has adopted a conformance testing procedure to ensure that each device conforms to the standard.

The conformance tests are performed by AEF accredited test laboratories. When a device passes the conformance tests it then receives an AEF ISOBUS Certification label which not only shows that the device conforms to ISO 11783 standard and AEF design guidelines but also the functionalities the device supports. [30] Figure 4.4 shows an example certification label.

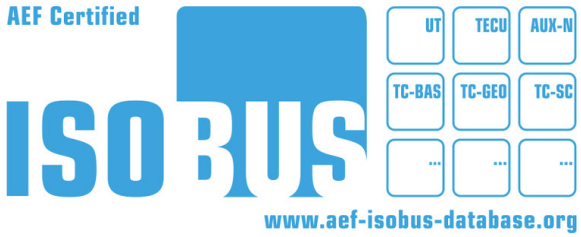

Figure 4.4: An example AEF ISOBUS Certification label. [30]

The boxes on the right side of the certification label show the functionalities the device supports. For instance the box on the upper left corner that is labelled "UT", means that the device can be operated using an Universal Terminal. This new certification label has just been adopted, and there are a lot of machines that use an older style label that does not list the supported functionalities. [30]

When a device gets the certification label, it can also be added to the AEF ISOBUS database. The database is mainly aimed for end-user so that they can easily get information about available certified ISOBUS devices and their functionalities. At the time of writing this thesis the ISOBUS database was not yet available for end-users. [30]

5. OTHER CAN-BASED HIGHER LAYER PROTOCOLS

In addition to the SAE J1939 and ISOBUS there are several other CAN-based higher layer protocols which define how the CAN data link layer is used in a specific application. Some of the most commonly used protocols are NMEA 2000, CANOpen, DeviceNet, CAN Kingdom and Smart Distributed System which are introduced briefly. Also their main differences compared to SAE J1939 and ISOBUS are presented.

Many of the other CAN-based protocols are designed to be used in different applications of industrial automation. This also means that the target environment for these protocols is very different when compared to SAE J1939 or ISOBUS which are used in moving machines. Usually the setup in industrial automation is relatively static and the network nodes are usually known at design time. In moving machines the network nodes can change when the network is operational, which also leads to very different requirements for the network protocol itself.

5.1 NMEA 2000

NMEA 2000 is a CAN-based higher layer protocol that is based on SAE J1939 similarly to ISOBUS. NMEA 2000 is designed to be used in maritime applications, and the standard series is developed by the National Marine Electronics Association (NMEA). It is used to interconnect navigation equipment, power generation and distribution systems, piloting and steering devices, alarm systems and other devices inside a ship. [33, p. 2-3]

The NMEA 2000 standard defines the physical layer including the connectors used in the network. Data link layer and network management are aligned with the ISO 11783 standard, and these parts are not fully defined by the NMEA 2000. Instead, the NMEA 2000 references ISO 11783 parts 3 and 5. Application layer is defined by the NMEA 2000 standard, because it defines the application field specific messages and the information that is exchanged in maritime systems. [33, p. 5-6]

Because the NMEA 2000 is based on SAE J1939 and ISOBUS, it is very close to them when it comes to its technical implementation. This means that the basic network mechanisms such as address claiming and how the CAN IDs are interpreted are same as in ISOBUS. However, the NMEA 2000 does define it's own connectors

and physical layer so these networks are not physically compatible.

Also, the application level information and functionalities are different because of the completely different application domain. In practise this means that the NMEA 2000 defines its own set of PGNs that describe the data objects needed in maritime communication whereas ISOBUS PGNs describe information found in tractor-implement combinations.

5.2 CANOpen

CANOpen is a CAN-based higher layer protocol for industrial automation systems. CANOpen was developed within CAN-in-Automation (CiA) members and it is standardized in GENELEC EN 50325-4 standard. [34]

Each device in the CANOpen network has an object dictionary which basically is an array of variables with 16-bit index and 8-bit sub-index. These object dictionary variables can be used to configure the device, and they can also reflect the device state by, for instance, containing measurement data. CANOpen standard defines a set of profiles which in turn define a set of object dictionary variables each device type needs to implement. [35, p. 28-30] This makes it easy to use devices from different manufacturers.

The CANOpen also defines a set of protocols for various operations in the network. For instance, Service Data Object (SDO) protocol is used to access entries in the object dictionary and Process Data Object (PDO) protocol is used for transferring process data between nodes. [35, p. 32-34]

When compared to ISOBUS and SAE J1939 the CANOpen has little in common with them. CANOpen is targeted for industrial automation where the system is usually very static. Therefore the original CANOpen does not have dynamic runtime address management as in ISOBUS [35, p. 318]. Also the application level is very different. CANOpen relies on device profiles which governs the structure of the object dictionary, whereas the functionalities in ISOBUS are mapped directly to PGNs and subsequently directly to CAN IDs.

5.3 DeviceNet

DeviceNet is a protocol which is designed mainly to be used between industrial controllers and I/O-devices (Input/Output), and it is therefore mainly used in industrial automation. DeviceNet is based on the OSI model, and it uses CAN for the data link layer and CIP (Common Industrial Protocol) for the higher layers. DeviceNet also provides power bus which makes it possible to use low-power devices without an external power-supply, thus reducing cabling. Open DeviceNet Vendor Association (ODVA) also provides conformance testing for DeviceNet products in a

similar fashion the AEF provides conformance testing for ISOBUS. [36, p. 1]

DeviceNet uses the standard CAN frames (11-bit identifier) for communication, and the extended format is not used. The CAN ID is used as a connection identifier, which in turn contains a MAC ID (media access code identifier) which is unique for each module. [36, p. 3-4] DeviceNet does not differentiate the device identity and address in a same way the ISOBUS does. Instead, DeviceNet uses the MAC ID directly in the CAN message to identify the device.

DeviceNet uses CIP for its upper layers. CIP is an object-oriented model in which each object has attributes (data), services (commands) and behaviour (reaction to events). The application data remains the same regardless of the lower layers which means that it is possible to connect DeviceNet network to another CIP-based network, for instance EtherNet/IP, without modifications to the application layer. Also, common application level profiles are used for different devices which makes it easy to use devices from different manufacturers. [36, p. 5-6]

Again, when comparing the DeviceNet protocol to SAE J1939 and ISOBUS the different requirements for the protocols are clearly visible. The DeviceNet is intended to be used in static environment so there is no dynamic address management (duplicate MAC ID check can be performed on start up). Furthermore, the object-oriented CIP application layer is very different when compared to SAE J1939 based application layer. DeviceNet also specifies standard device profiles in a similar fashion as CANOpen.

5.4 CAN Kingdom

CAN Kingdom differs from the other CAN based protocols presented so far in a one fundamental way: there is a master node which controls everything. When most other networks are open in the sense that a module can begin to transmit immediately when it joins the bus, a new module in CAN Kingdom is only allowed to wait instructions from the King. However, the King node is only needed in the configuration phase. When the system has been configured and each device has stored its configuration to a non-volatile memory, the King can even be removed. [37]

CAN Kingdom differentiates module design and system design. This gives the system designer a lot of freedom to configure, for instance, the network topology and priorities. For example in SAE J1939 and ISOBUS the protocol group number affects the CAN ID, which in turn is used for prioritization. In CAN Kingdom the message priorities can be selected by the system designer. [38, p. 1]

In CAN Kingdom all the needed nodes and connections between them must be known in the design time so that the King can configure them in the configuration phase. This approach works in control systems, where the system configuration is

static during runtime. [38, p. 1] However, in ISOBUS system the user can connect or disconnect devices from the network runtime, and therefore this approach is really not applicable for agricultural environment.

5.5 Smart Distributed System

Smart Distributed System (SDS) is a CAN-based higher layer protocol designed by Honeywell for industrial automation needs. It is designed to be used between industrial controllers such as PLC's (Programmable Logic Controller) and IO-devices like sensors and switches. Fundamentally SDS is a point-to-point protocol between master device and slaves, which makes it effective for its intended use. [37]

Similarly to CAN Kingdom the SDS also assumes that there is always at least one host node in the system, that can perform validity checks and configure the slaves if needed. However, the SDS does not limit the number of master devices to a single instance in a way the CAN Kingdom does. [37] SDS is designed in a fundamentally different way compared to ISOBUS where no predefined master/slave roles exists.

SDS uses either 125k, 250k, 500k or 1M baud-rate which is selected by the master devices. SDS also defines "autobauding" procedure which makes it possible to automatically configure the baud-rate for slaves. In SDS network only standard CAN-frames with 11-bit identifier field are used which is also a major difference when comparing to ISOBUS. Similarly to DeviceNet, SDS also defines profiles for different device types including behaviour and data structures to and from the devices. [37]

6. TELEMATICS SYSTEM ARCHITECTURE

In this chapter the requirements for telematics system, and its high level architecture is discussed. The main emphasis is on the component level and interfaces, so therefore the underlying hardware and software components which implement these interfaces are not addressed in detail.

Chapter 6.1 first goes through requirements which were identified by the usage examples described earlier in Chapter 2. Thereafter, Chapter 6.2 provides an example reference architecture that fulfils these requirements. Followed by the architectural discussion the information available in ISOBUS systems in general is presented in Chapter 6.3. Then the representation and binding of the machine data is discussed in the Chapter 6.4. Finally, Chapter 6.5 analyses the available interfaces and their applicability to telematics.

6.1 System requirements

The requirements presented in this chapter are based on the earlier usage examples in Chapter 2 and also discussions with Finnish agricultural machine manufacturers in the Agromassi project. The main requirements for a telematics system are to provide a means to gain remote access to a machine in its working environment as well as to record information from it to a remote storage outside the machine.

The device that connects to the ISOBUS network in the machine needs to be able to connect to Internet wirelessly. This way the machine can be accessed remotely in its working environment. The system also needs to be able to collect information from multiple machines simultaneously, and the collected data should also be accessible when the machine is offline. The system should also provide previously collected history data. Furthermore, both a user-interface and software interfaces for external systems are required.

There are also requirements that are not strictly necessary for the core functionality, but are needed to make the system usable, maintainable and safe. One of the most important design aspects is modularity and a proper use of interfaces to abstract the modules. Modular design makes the system easier to maintain and also makes it possible to extend the system in future. Modularity in the telematics system design is addressed in the following architecture chapter.

Another important aspect is safety. In the telematics system the on-board device

in the machine is connected to the machine's internal communications network. As described in the Chapter 4, the ISOBUS network is used, for instance, for the communication between the Universal Terminal and implements. Special care needs to be taken so that the telematics system does not interfere with other devices on the bus. The other parts of the telematics system are not in a direct contact with the machine, and thus not safety-critical.

There are also boundary conditions which affect how the system is designed and how it can be used. Most notable boundary conditions are the limitations caused by the mobile network (wireless Internet connection). The non-deterministic network delays and possible random connection failures in the wireless mobile network means that the telematics system should not be used for machine control. The system also needs to be designed in a way that no data is lost even if the connection is lost temporarily.

6.2 System architecture

It is a common situation in an agricultural environment that there are more than one tractor-machine working-set that should be monitored simultaneously. This is why the telematics system should be distributed so that telematics devices are placed to individual machines, and they connect to a shared server system. Figure 6.1 shows how the components in the telematics system could be distributed.

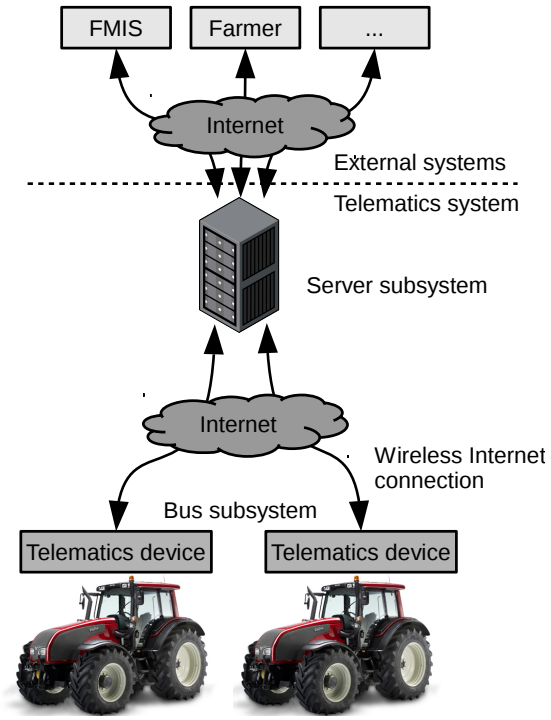

Figure 6.1: Distribution of the telematics system.

The figure shows one server system and two telematics devices that are installed in tractors. The telematics devices are connected to the server system using wireless Internet connection. The system end-users and other software systems would connect to the server, and not directly to individual telematics devices. This approach allows flexibility on the hardware platforms used. The telematics devices could be implemented using an embedded platform, whereas the server-side can use a generic server hardware or it can be implemented as a cloud service. This also improves the system scalability.

An example reference design that fulfils the requirements presented in the previous chapter and also supports distribution is shown in the figure 6.2. The system consists of two main subsystems which differ significantly from each other. These subsystems are the bus-side subsystem which connects to the actual communications bus in the machine and the server-side subsystem which stores the data and offers interfaces outside. There is also a communication links between these two subsystems and between the server and the outside world.

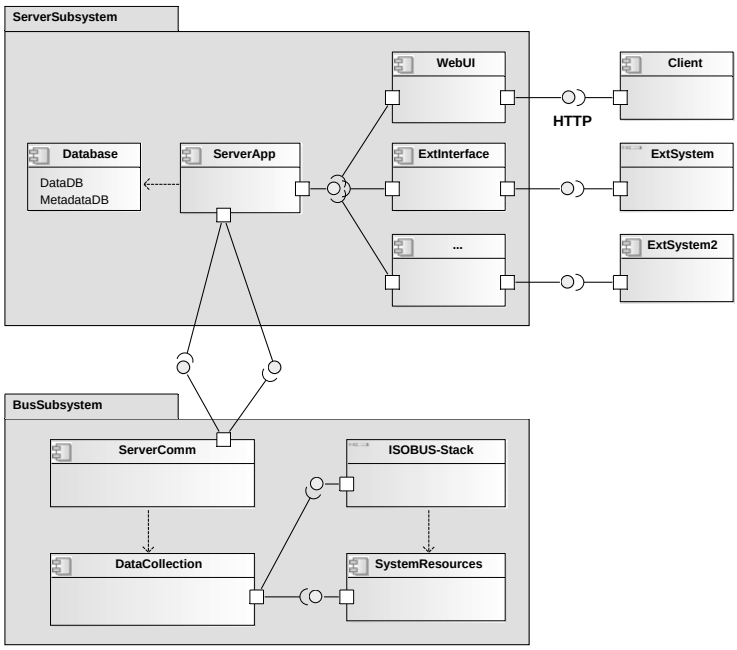

Figure 6.2: Telematics system component diagram.

The two main subsystems are shown as packages, and for both packages some of the most important internal components are also shown. In addition to the the actual subsystems, external systems that could utilize the telematics system interfaces are also shown on the right side (*Client*, *ExtSystem*, *ExtSystem2*). These external systems correspond to the FMIS and Farmer in the figure 6.1. The architecture presented here is a system-level architecture, and each of the shown components may have complex internal architecture as well.

The server subsystem consists of database component (*Database*), the actual server application that implements the application logic (*ServerApp*) and components that implement different interfaces for external systems or clients (*WebUI* and *ExtInterface*). These external interfaces could be for instance REST, OPC UA, SOAP and others. The *ServerApp* component manages the remote devices, authenticates them and stores the actual process data to database. It also provides internal interfaces to access the data and to manage the remote devices. The server-side subsystem is described in detail in chapter 6.2.2.

The bus-side subsystem, or the remote device, consist of software component that handles the communication with the server subsystem (*ServerComm*), application logic that initiates and performs the actual data collection (*DataCollection*), ISOBUS-stack that handles the protocol specific communication and routines (*ISOBUS-stack*) and finally system resources such as device drivers (*SystemResources*). The bus-side subsystem is described in detail in chapter 6.2.1.

The component diagram also shows the main abstractions in the telematics system. The interfaces between the server-side subsystem and the bus-side subsystem creates an abstraction that separates the two subsystem from each other. This way the internal implementation of the telematics devices or their hardware platform does not affect the server. Second distinct abstraction are the interfaces to the outside world. For an external system, the telematics system in whole is seen as a black-box.

The separation between the bus-side system (*BusSubsystem*) and the server-side system (*ServerSubsystem*) is a key design decision in this architecture and it allows these two subsystems to be distributed in different physical computer systems as shown in the figure 6.1. Communication between the subsystems could be implemented using, for instance, mobile networks, and various communication protocols can also be used. For instance, binary protocols for size optimized communication or OPC UA or Web Service interfaces for more generic approach.

6.2.1 Bus-side subsystem

Bus-side subsystem implements the link between the telematics unit and the communications bus of the machine. It is also responsible for providing the configured data to the server-side of the system. The software that implements the bus-side subsystem functionalities is usually run on an embedded platform that is designed for harsh environments, and in agricultural applications could be placed, for instance, in tractor. The figure 6.3 shows an example structure of a complete software stack for an ISOBUS enabled embedded device. The software that was shown in the component diagram in figure 6.2 is located on the topmost layers of the software stack.

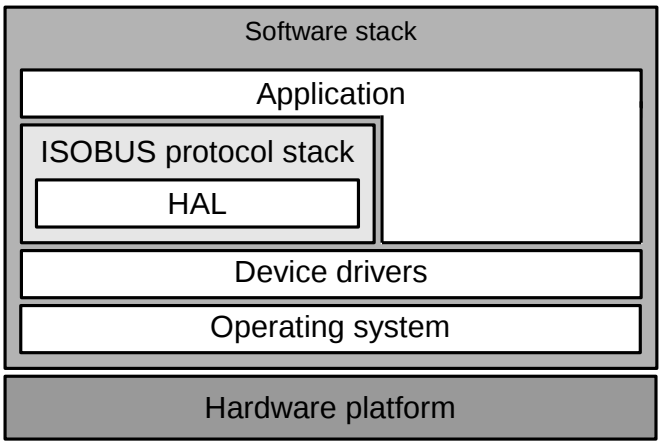

Figure 6.3: Software stack of ISOBUS enabled remote management device

On the very bottom is the physical hardware platform. On top of the hardware platform is the operating system such as embedded Linux, Windows or a RTOS, and also device drivers that give the application level access to the underlying hardware. It is also possible to implement the embedded platform without a generic operating system. However, the operating system creates a distinct abstraction between the application and the hardware which improves the portability of the system, and can also simplify the system design.

To make the application more maintainable and to improve re-usability, it is also a good practise to separate the actual application and the ISOBUS protocol stack. This way the protocol stack can be developed as an individual module that can be re-used. Inside the protocol stack it is also a good practice to have a separate Hardware Abstraction Layer (HAL) that contains all the parts that are hardware or driver dependant. This way the protocol stack itself can be more easily ported to another hardware.

It would also be possible, that the external systems would connect directly to the telematics devices without any server system (both *BusSubsystem* and *ServerSubsystem* would be placed in a single physical device). However, this introduces many problems that are avoided when using a separate distributed server. First of all, the remote device is powered from the machine, and therefore it is only available when the machine is used. Secondly, if many users want to access the device simultaneously, both network bandwidth and the device performance may not be sufficient. Also the storage capabilities for embedded devices are usually limited. When the device communicates with a single server which in turn provides the external interfaces and data storage, these problems are avoided.

6.2.2 Server-side subsystem

The server-side subsystem is responsible for collecting and storing data from the remote telematics devices (*BusSubsystem*) on the machines. It is also the side of the system that is visible to the telematics system clients whether they are end-users or other software systems. The software that implements the server-side subsystem is usually run on a general purpose platforms that do not require any special customizations for use in ISOBUS telematics. All the low level dependencies caused by ISOBUS are abstracted by the bus-side subsystem.

The figure 6.2 showed some of the most important internal components in the server-side subsystem. An important component is the database component that abstracts the actual database accesses and provides the main server application with storage functionalities. In the telematics system there are various kinds of data that needs to be stored. First of all, the actual measurement data and also user access information as well as device configurations.

The main application logic in the server system is implemented by the *ServerApp* component. It aggregates the data from multiple remote devices and stores it to database. It is also responsible for authenticating the remote devices and for providing them with a valid configuration.

To make the server subsystem more modular there are also internal abstractions. The *ServerApp* component provides well-defined interfaces to access it's resources. This way there can be multiple different interfaces and modules that use the server resources but do not need to know about it's internal implementation. The component diagram shows, as an example, *WebUI* and *ExtInterface* components which use the same internal interfaces to access the resources.

User interfaces provide means for humans to interact with the system. There are various different kinds of users, such as developers, administrators and end-users, so usually multiple different user interfaces are needed. Alternatively, the different user types may be presented with a different view or perspective within the same user interface. In the component diagram a web based user interface was shown (*WebUI*), but also other interfaces could be used.

In addition to the user interface also software interfaces can be provided. Software interfaces are not meant to be used directly by end-users but instead by other machines or systems. These software interfaces expose the telematics system resources and make them available for external systems. These external systems can then implement other applications and provide user interfaces for the end-users. In the component diagram *ExtInterface* represents a concrete implementation for one external interface, and *ExtSystem* a third party system that accesses the telematics system.

6.3 Information available in ISOBUS network

A high-level requirement for the telematics system is to be able to collect and record data from the target machine. ISOBUS supports proprietary communication so in principle arbitrary information can be collected from an individual machine. However, when designing a generic telematics system, the use of proprietary messages will reduce its applicability. Therefore a better approach is to use the methods provided by ISO 11783 standard as much as possible. This chapter presents the results for literature and ISO 11783 standard study of available information sources that was done in this thesis. The ISO 11783 standard does not address remote telematics, so the standard is studied from the telematics point of view to identify which existing data sources and interfaces could be adapted for telematics purposes.

There are two main methods for collecting data from a vehicle communications bus. The device collecting data can either passively listen to the communication and store interesting data, or it can actively be part of that communication. The former method may be easier to implement, but it also has significant constraints because the device can not query data actively. The latter method, in turn, allows the device to work within the limits of the communications protocol. Table 6.1 summarizes the different types of data that can be collected from an ISOBUS network.

Table 6.1: Data collection methods.

	Passive monitoring	**Active communication**
Proprietary	• Network statistics • Proprietary broadcast messages	• Proprietary peer to peer protocol (arbitrary information can be transferred)
ISO 11783	• Broadcast ISOBUS PGNs • Broadcast active diagnostics trouble codes	• Address claim information • Request PGNs • ISOBUS diagnostics protocol • Process data logging • File server

Passive monitoring means that the telematics device connects to the communications bus, but does not send anything. In an ISOBUS system this would mean that the telematics device would not, for instance, do an address claim. When using passive monitoring the telematics device just listens to the CAN messages transmitted to the bus and filters the interesting messages by using the message's CAN ID. This

method could be used to collect both proprietary and ISOBUS specific broadcast messages that are automatically sent to the bus by some other device. From the implementation point of view, this method is very simple because it does not require a higher layer protocol stack. Simple CAN access is sufficient. In addition to the actual CAN messages, it is also possible to measure network statistics such as bus load, voltages, message errors or physical bus failures.

The proprietary broadcast communication can be performed using base format CAN-frames because the ISOBUS only uses extended frame format [27, p. 2]. However, when using the base frame format, there is a possibility for CAN ID collisions between manufacturers. Therefore the ISOBUS also provides a PGN for proprietary broadcast communication [27, p. 17-18]. When using this message type, the sending device must have an address claimed. From the telematics point of view this method is not very interesting because it relies on proprietary manufacturer specified messages.

In spite of its simplicity, the passive monitoring is a useful method to collect data from an ISOBUS network. A useful data source for monitoring is the tractor ECU which broadcasts a lot of information to the bus automatically. For instance, machine speed, hitch positions, power take-off rotation speed and much more may be broadcast. These messages are specified in detail in ISO 11783 part 7 [28]. AEF defines two different levels of functionality for tractor ECUs. TECU Basic provides only a limited amount of information, and does not support bi-directional communication. TECU Advanced broadcasts more information and also supports command requests from implements. [39] Controlling the tractor is not an important functionality for telematics, but the additional information provided by TECU Advanced is useful. If the ISOBUS support has been retrofitted to the tractor, there might not be ISOBUS capable tractor ECU at all.

Another useful information that is automatically broadcast to the bus is active diagnostics trouble codes (DTC). When a fault condition becomes active, an ECU starts sending Diagnostics Message 1 (DM1) periodically which contains information about the fault. The message contains SPN for the subsystem that had failed, and also Fault Mode Indicator (FMI) that identifies the type of failure. This message is specified in detail in ISO 11783 part 12. [40, p. 10-11]

If the data obtainable from passive monitoring is too limited, the next step would be to start actively communicating on the bus. Again, the base format CAN frames can be used for peer-to-peer communication. This method does not necessarily require a higher layer protocol stack. ISOBUS also provides a PGN for peer-to-peer proprietary communication [27, p. 17-18]. This however, requires that both devices have an address claimed. These methods only allow manufacturer specific communication, and therefore are not a good approach for a generic telematics

system anyway.

To get most information from the ISOBUS network, the telematics device should have an ISOBUS protocol stack so that it can utilize the full potential of the protocol. This is also the option that was used in the example reference design because it allows much more information to be collected than passive monitoring. All of the data sources that are visible in the Table 6.1 become available when using a ISOBUS protocol stack.

First operations to do when a device joins the ISOBUS network is to claim an address. This operation alone provides information about other devices on the network and their ISOBUS NAMEs. The NAME in turn contains information about the device type and its manufacturer. So, just by joining the network, the telematics device can obtain valuable additional information compared to passive monitoring. It is worth noting that the "Address Claim" messages are broadcast and can in theory be obtained only by listening. These messages are not, however, sent automatically if no-one first sends an "Request For Address Claim" message.

The ISO 11783 defines a set of message PGNs that are used to request a message with a specified PGN to be sent. These requests can be both global or directed to a single device. This method allows the telematics device to query for information that is not automatically broadcast. [27, p. 13-14] It is, of course, not guaranteed that the requested information will be available. For instance, "ISOBUS compliance certification message" could be queried to get information about the device ISOBUS conformance testing or "ECU software identification" to get additional information about the ECU firmware.

The ISO 11783 part 12 also defines additional diagnostics functionalities in addition to the DM1 message. It is possible to, for instance, request previously active diagnostics trouble codes. Also, the "ECU diagnostic protocol" message allows to query if the device supports other diagnostics protocols in addition to the methods defined in ISO 11783. If the ECU supports them also SAE J1939-73, ISO 14230 (KWP200) or ISO 15765-3 (Unified diagnostics services on CAN) can be used to obtain additional information. [40]

Apart from the broadcast PGN messages, the data sources discussed so far have been mainly diagnostics. However, it is also possible to obtain process data from implements during their work assignments. The ISO 11783 part 10 defines a generic data logging interface that can be used to monitor the process variables during work assignments. This is is a similar interface the task controller uses to monitor tasks. The available process variables are defined by Device Description Data -file (DDD-file) that is provided by ECUs that support task controller. The data logging is a separate functionality, and all implements do not support it even though they support the task controller.

The ISO 11783 also defines a File Server functionality that allows other ECUs to read and write files to a common storage. If the telematics device implements the file server functionality, it could be used to make external files available to the ISOBUS network. For instance, upload configurations or tasks from server to ISOBUS file server or send log files written by ECUs to the server. These functionalities require that the ECUs use the file server, so they are not completely manufacturer independent, but at least the proprietary messaging can be avoided.

6.4 Data representation

Raw data by itself has little value without a context to tell what the data represents. Therefore, when data is collected, it is also necessary to store meta-data to preserve the information value of this data. If the connection between the data values and the context is lost, the data becomes useless. It is also important to know where the collected information originated.

Usually a system has a data model that explicitly defines the structuring of data objects within the system. This data model is constructed in a way that it allows to represent information about the real-life objects that the information represents. So basically the data model defines the different entities that can exist as well as allowed connections between them.

The data model defines the logical structuring of data objects. However, this information also needs to stored and communicated which means that it needs to have a well-defined representation. From the ISOBUS point of view, the data representation on the bus is governed by the ISO 11783 standard series which specifies the CAN communications in detail. The standard also defines XML-bindings for information needed to transfer ISOBUS tasks between the task controller and FMIS.

As a part of the FMIS study by Robbenmond et al. also the data exchange between FMIS and other actors was studied. The study concludes that "there is a lack of consensus about used data standards for data exchange in agriculture". They also conclude that "no standardised messages from international standardisation organisations have been discovered that facilitates data exchange between sensors and the FMIS or data providers and the FMIS". [3, p. 27] Therefore, from the telematics point of view it is not apparent which data model or data representation format should be used.

There are attempts to provide a general data model and representations for the agricultural domain. For instance, agroXML project provides a set of XML schemas for representing data on work processes on the farm including accompanying operating supplies like fertilizers, pesticides, crops and the like [41]. Still, the current situation is that there are many different data models in use in the industry.

Another approach that is applicable for the telematics system is to use a more generic data model. In this approach the generic entities are mapped to actual domain specific objects when to telematics system is integrated to other systems. This approach requires additional translations between the data models and can therefore reduce performance. However, because of the various data models and data formats that are used in agriculture, this kind of translations are likely to be needed anyway.

6.5 Design considerations

The state-of-the-art study in Chapter 2.2 showed that the vast majority of the ECUs in the current agricultural machine base do not have the required hardware to connect to Internet. This means that the telematics functionalities in the current machines would need to be implemented using a separate device that provides the Internet connectivity. In future these functionalities could be integrated to other ISOBUS devices such as tractor ECU, implement ECU or Universal Terminal if the required hardware support is added. These devices could then communicate directly with the telematics server.

The results for the study of available data sources and interfaces in an ISOBUS system was presented in Chapter 6.3. These results show that a telematics device can collect versatile information ranging from diagnostics to actual process data in a manufacturer independent way. The study also shows that the telematics system could be provided by a third party provider independently of the actual machine manufacturer, and the system could be installed to current machines as and add-on. Furthermore, the study suggests that the telematics part for all the usage examples presented in Chapter 2.3 could be implemented without resorting to proprietary communication.

The file server interface in particular provides very interesting possibilities for telematics. The files transferred to the file server by other ISOBUS devices could be uploaded to the server subsystem. This means that, for instance, system logs, test results or basically arbitrary data could be transferred from a machine without using any proprietary protocols. For example, in malfunction situation the support personnel could obtain very detailed logs from the machine.

The files could also be downloaded from the server to the telematics device to provide, for instance, firmware updates to machines that do not have Internet connectivity. The file upload and download could also be used to transfer tasks between the on-board task controller and FMIS to remove the manual transfer. It would be beneficial if the use of file server would become a common practice for firmware updates and task transfer instead of proprietary solutions for each manufacturer. This would also serve the original purpose of the file server standard, and benefit

both the machine manufacturers as well as telematics system providers. The main limitation for using the file server to transfer large files is the limited bandwidth in an ISOBUS network (250 kbit/s). Also, only a fraction of the maximum transfer rate could be used for file transfer so that other communication is not disrupted.

Another very interesting interface is the process data logging interface which exposes the actual controllable variables in a machine which could be monitored in real-time. In precision farming the process variables are used to control the implement based on location to optimize the work. The process logging and task controller interfaces also support read-only variables which means that other information could also be exposed using this interface; individual sensor readings for instance. Also, the broadcast messages from tractor ECU would be interesting for remote monitoring, and also to make available in FMIS.

From the implementation point of view the diagnostics, file server and process data logging are well-defined by the ISO 11783 standard. Therefore the actual software architecture and implementation can vary, but the interfaces used are standardized. However, as Chapter 6.4 showed, there is no consensus about the general data model or the data exchange formats in agriculture. Therefore it is likely that various different data models and data formats are encountered when the system is integrated to existing FMISs. This is a major challenge that should be taken into account in the system design.

Based on the identified usage examples in Chapter 2.3 and available data presented in Chapter 6.3 three telematics functionalities were selected for implementation in the Agromassi project. First, collection of ISOBUS frames based on their PGN allows to monitor the information sent by the tractor ECU. This functionality can be used for remote monitoring, and it can also be integrated to FMIS. Secondly, a device discovery functionality and the collection of active diagnostics trouble codes allows the system to support basic remote fault diagnostics. Lastly, file server functionality was selected because it allows to transfer versatile data, including firmware update packages and tasks, between devices on the bus and external systems. These functionalities were selected because they allow the system to support basic applications for all of the identified usage examples.

This chapter provided a high level example reference architecture which can be used as a starting point for ISOBUS telematics implementation. Also, the different data sources in an ISOBUS system and their applicability were assessed. The following chapters show how the ISOBUS support as well as telematics functionalities were added to the WRM system.

7. WAPICE REMOTE MANAGEMENT SYSTEM

Wapice Ltd is a Finnish company that concentrates on software solutions and integration of information systems of industrial companies. The company was founded in Vaasa Finland, and today it employs over 260 software and hardware experts (situation on spring 2014). Wapice is also ISO 9001:2008 certified. The company is grouped to three segments which are embedded systems, industrial systems and business solutions. In addition to subcontracting, Wapice also develops it's own solutions including Wapice Remote Management system. [42]

This chapter gives a general introduction to the WRM system in which the ISOBUS support was implemented in this thesis, and which was also used during the Agromassi project. First, the WRM system concept and its most important parts are discussed. Then the remote management devices in WRM system as well as the server side of the system are introduced.

7.1 WRM concept

The WRM-system is a complete solution which consists of remote devices, server system and software. The system provides access to machines and devices regardless of their location, which makes it possible to integrate them to other information systems. The remote devices are located in the field and they connect directly to target systems using, for instance, field-buses. The server system collects the data from remote devices and stores it to database. It also provides user interfaces to configure devices and measurements as well as software interfaces for external systems. Figure 7.1 shows the WRM-system concept. [43]

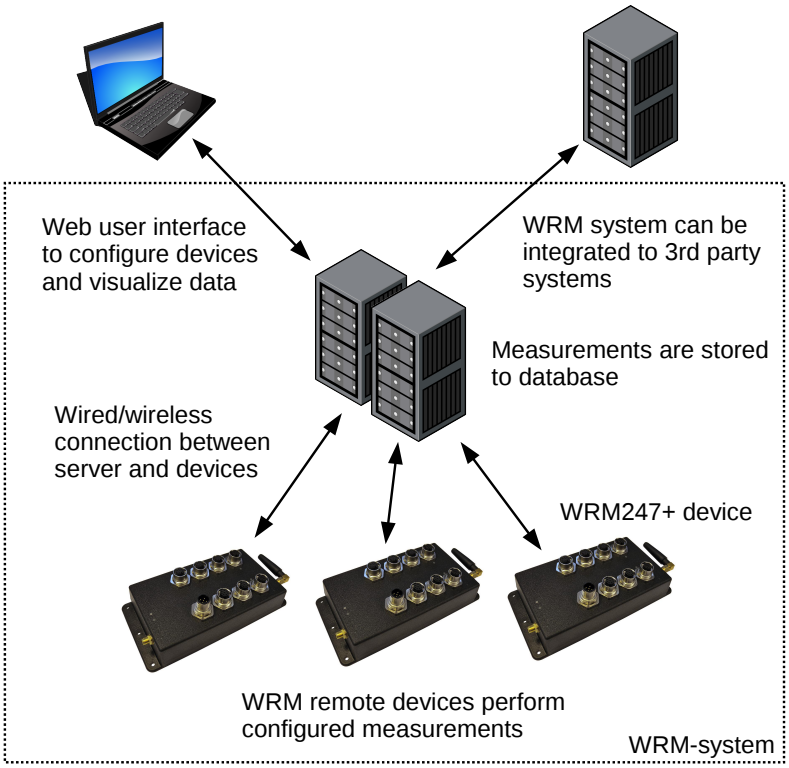

Figure 7.1: WRM system concept.

The dashed line in the figure shows the WRM-system boundary. Inside the WRM-system a number of remote management devices and the server system are shown. The remote devices connect to the server via Internet using either wireless or wired connection. The server system also contains databases for the actual measurement data, user information as well as the remote device configurations. The server also provides Web user interfaces to manage the devices and visualize the collected data. Software interfaces are also provided for external systems.

7.2 Remote management device

The concept figure also shows an example remote management device, WRM247+, that can be used in the WRM-system. The WRM247+ is a cost-efficient device that is designed to meet industrial requirements, and to support a wide range of common interfaces. It can be connected to the WRM servers using either wired or wireless technologies. Especially the possibility to use mobile networks (GSM/GPRS/3G) makes the device suitable for moving machines including agricultural applications. Table 7.1 lists some of the WRM247+ devices' technical details.

Table 7.1: WRM247+ technical details [44].

Feature	Description
Processor	ARM® Cortex-A5, 536MHz
RAM	128MB DDR2
Flash	256MB NAND Flash
Connectivity	Ethernet, CAN, RS-232, RS-485, USB
Wireless	GSM/GPRS/3G, WLAN and Bluetooth
Positioning	GPS+GLONASS, -165dBm sensitivity, 33 channels
Input/Output	4x Digital In, 4x Digital Out, 2x Analog In
Accelerometer	Digital, triaxial, 16 bit, ±2g - ±16g
Operating system	Real-time Linux
IP Rating	IP 65

The WRM247+ device uses embedded RT-Linux as its operating system, on top of which the Wapice's own software solution has been implemented. The software handles the communication with the server and performs the configured measurements. The goal of this thesis is to add ISOBUS support to the WRM remote device's software which makes it possible to use WRM as an telematics system in agricultural applications.

7.3 WRM server

Another important part in the WRM system is the server. The main responsibilities of the server system are managing device configurations, storing the data received from remote devices, providing user interfaces and access control. Normally the remote management devices are installed on various automation systems, so the server system is the side that is really visible to the system users.

The WRM-system users can view, modify or add measurements to the server data model using the Web-interface. The server also supports user groups so that the data model visibility and access rights can be controlled. Furthermore, the WRM-server instances can be run in virtual environment or deployed directly to customer servers to enhance information security. The server data model can also be accessed using the REST (Representational State Transfer) interface.

The REST interface allows the WRM system to be integrated to third party systems. Currently the REST interface supports browsing the data model, reading history data from measurements and online read/write of data nodes. However the interface does not allow, for instance, to create new data nodes or configure the remote devices. These are features that could be added in future on per need basis. The REST interface was also used in the Agromassi project to access the collected data.

The WRM data model is hierarchical and it is not domain specific so that it can be used to model various real-life applications. On top of the hierarchy are enterprise nodes which represent companies or other organizations. Enterprises consist of multiple sites which are usually physical locations such as factories, which in turn consist of assets. Each asset can have multiple data nodes which represent individual measurements. There are multiple different types of data nodes which defines how the data is obtained, for instance CAN message or I/O value. [45] Appendix A presents the WRM data model using class diagram notation, and also an object diagram that illustrates an example modelled using the WRM data model.

8. ISOBUS SUPPORT IN WRM SYSTEM

During the Agromassi project ISOBUS support was implemented to the WRM-system. This makes it possible to use the WRM-system as an infrastructure for telematics functionalities described in the earlier chapters.

The server side of the WRM-system was used as is, and the implementation needed for ISOBUS support was done on the remote device's software. This follows the architectural idea introduced in the Chapter 6 that the ISOBUS specific implementation is abstracted to the remote devices. This chapter describes the ISOBUS functionalities and the general ISOBUS support implementation that was done to the WRM remote devices.

8.1 Telematics functionalities

In this thesis two concrete telematics functionalities were implemented to the WRM system. These functionalities highlight the different requirements already discussed in the chapter 6.1. They also ensure that the whole chain from the physical bus to the server interfaces is functional. During the Agromassi project a file server implementation was also planned, but due to resource cuts from other partners, this functionality was omitted from the respective work package.

The first functionality is the collection of raw ISOBUS frames according to their protocol group number. This functionality allows the user to configure arbitrary amount of message PGNs which are then automatically collected. It is also possible to configure the actual bytes from the CAN data which are kept as well as scaling. This functionality is mapped to the WRM data model as a new data node type. In practice, this means that the collected data can be directly used in WRM user interface widgets such as charts or gauges. The data is also available using the WRM REST interface. This functionality can be used to collect broadcast messages from tractor ECU for instance to support basic performance monitoring.

The process data logging interface would allow to monitor the same variables the task controller controls. However, this functionality was not selected for implementation because of its limited support in ISOBUS devices. The lack of support was also verified for an open-source ISOBUS stack, ISOAgLib, during the Agromassi project. In future this functionality could also be implemented to extend the data available for monitoring.

The second implemented functionality is device discovery which allows to list all the devices that have claimed an address and thus registered to the bus. From each of the devices their claimed bus address as well as their ISOBUS NAME can be obtained. The address claim procedure was introduced in Chapter 3.2. Additionally, information about the device's active diagnostics trouble codes is recorded. From each active DTC, their failure mode indicator (FMI) and occurrence count is also saved. This functionality can be manually triggered from the server which makes it suitable for online troubleshooting. The device discovery can also be triggered using the REST interface. Again, this functionality maps to WRM data model as a new data node type.

The file server was implemented to a point where the WRM remote device could provide file handling to other ISOBUS devices according to the ISO 11783 part 13. This means that other devices on the bus could read and write files as well as handle directories. Also, manufacturer specific directories were implemented which are visible only to the devices of the same manufacturer (according to their ISOBUS NAME) [31]. However, this functionality was neither integrated to the WRM main software nor added to the server because it was omitted from the respective work package.

The implementation uses the data model that is already available in the WRM system. This is due to the fact that there is no consensus about the data models or formats in agriculture so adaptation to various systems is needed anyway when systems are integrated. Also, using the pre-existing data model simplifies the implementation.

8.2 CAN-drivers and protocol stack

An important aspect about the remote device from the ISOBUS point of view is its connection to the CAN-bus. In the WRM-devices the CAN-drivers are custom implementation which allows to optimize their performance for the specific hardware. Junnila et al. describe the optimizations done in the Wapice Custom CAN (WCC) driver as well as performance comparisons in their article [47]. These optimizations include for instance preprocessor branch hints and use of direct memory access (DMA). In summary, the WCC-driver offers significantly better performance in the WRM-devices when compared to other common Linux CAN-drivers such as SocketCAN or LinCAN.

To make the application development easier and also abstract the device driver usage, a library component is used to provide higher level interface to use the CAN-functionalities. This component is *LowLevelCAN*-library (LLC) and it is also developed in Wapice. the LLC-library provides interfaces to configure CAN-channels as well as to receive and send messages through these channels. The LLC-library

uses singleton design pattern which means that there is one shared instance of it that manages all the CAN-channels and their parameters. The LLC-library also uses observer design pattern which means that other modules can register listeners and receive automatic notifications when new CAN-frames are received. Figure 8.1 shows an overview how the LLC-library is used.

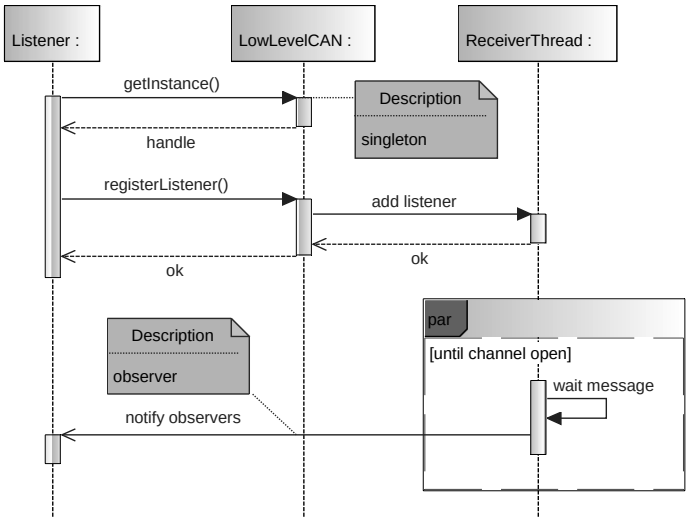

Figure 8.1: Using the LowLevelCAN library.

The WCC-driver accompanied with the LLC-library provides a means to handle basic CAN communication. However, to use ISOBUS functionalities, the actual ISOBUS protocol support needs to be implemented using these components. The ISOBUS protocol specific functionalities could be directly implemented to the WRM application, but a better design approach is to use a separate protocol stack which not only abstracts a lot of the ISOBUS specific routines but also makes the application level implementation easier.

The protocol stack could be implemented from ground up, but to reduce implementation effort an existing protocol stack was used. ISOAgLib was selected as the ISOBUS protocol stack [46]. The arguments for selecting ISOAgLib were suitable licence, active development and experiences from previous projects in-house. It is also constructed in a modular way, so that only the necessary functionalities can be compiled to the stack. For instance, universal terminal interface is not needed in the WRM so this functionality can be omitted from the stack. On the other hand, new features can be added in a modular fashion in future. Figure 8.2 shows the architecture of the ISOAgLib stack.

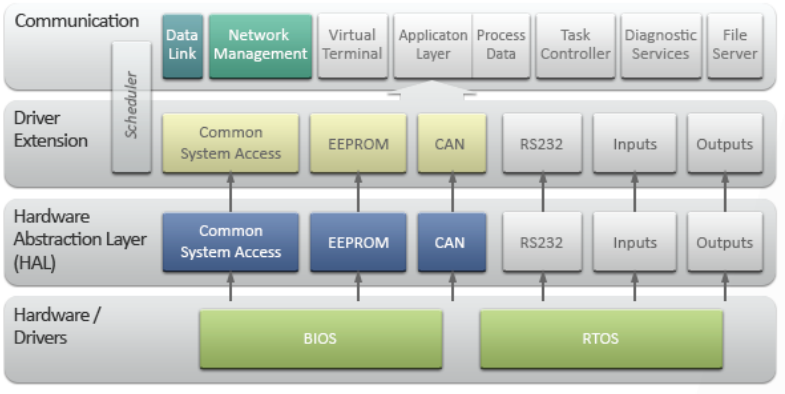

Figure 8.2: ISOAgLib protocol stack architecture. [48]

The layered architecture shows hardware resources provided by the operating system on the bottom. The bottom layers of the actual protocol stack are the hardware abstraction layer that provides access to the system resources and also driver extension. The top layer is the actual ISOBUS stack that consists of common data- and network management handlers as well as modular functionality blocks such as Task Controller client.

To integrate the ISOAgLib to WRM-device, a hardware abstraction layer (HAL) needed to be implemented so that the protocol stack can access the device resources such as CAN. The ISOAgLib specifies interfaces for CAN access and system resources which abstracts the protocol stack from the underlying hardware. The ISOAgLib can be ported to a new platform by providing device specific implementation for these interfaces. Overview for the HAL implementation using LLC-library is shown in Appendix B.

8.3 Integration to WRM application

The WRM remote devices support a range of protocols that are common in the industrial applications. This means that the ISOBUS functionality may not be needed at all in the majority of the applications. Therefore the ISOBUS stack or other ISOBUS related functionalities should not be consuming system resources when they are not needed. This is also true other way around. When the ISOBUS functionality is used, other unnecessary functionalities should not consume the resources. This is a key factor in the software architecture of the WRM system.

In the WRM system the ISOBUS functionalities are used in a different manner than in, for instance, implement ECU. In an implement ECU the ISOBUS plays an important role, and most of the functionalities are built around it. Whereas in the WRM system, the ISOBUS is just another data sources among others. This distinction also reflects to the software implementation which contains configurable

modules for various data sources. The general software stack of the WRM remote management devices is shown in the figure 8.3

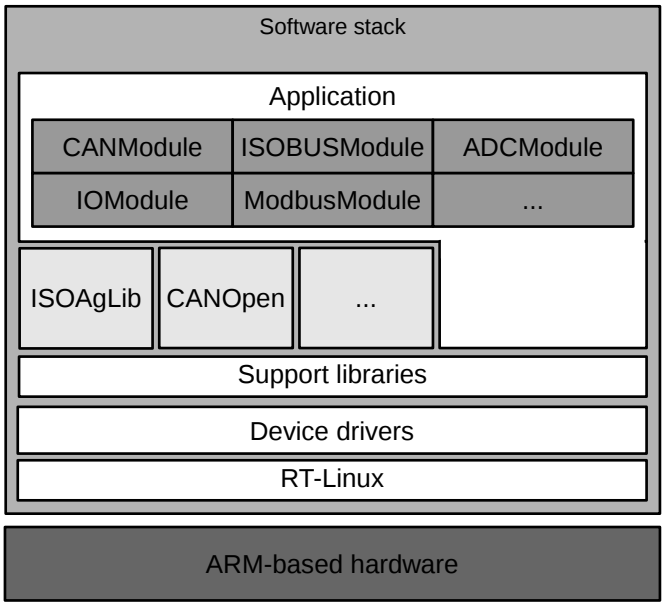

Figure 8.3: Software stack of the WRM remote management device.

At the bottom of the figure is the ARM-based hardware platform. One example for the hardware, WRM247+, was already introduced in the chapter 7.1. On top of the hardware is the operating system which in WRM devices is an embedded real-time Linux. On top of the operating system are the device drivers that provide access to the underlying hardware such as CAN-controller. To make the application development easier, there are support libraries on top of the device drivers, *LowLevelCAN* for instance, which simplify the device access from a high level application.

The software stack also shows that the ISOAgLib protocol stack has been separated from the actual application, and it is an individual software module. The ISOBUS functionalities in the application are implemented by the *ISOBUSModule* that uses the resources provided by the protocol stack. To minimize the dependencies to the protocol stack, the ISOAgLib is only accessed from within the *ISOBUSModule*. Also, there are abstractions inside the *ISOBUSModule* that hides the actual protocol stack usage from the rest of the application. Figure 8.4 shows a concept class diagram of the ISOAgLib integration to WRM application.

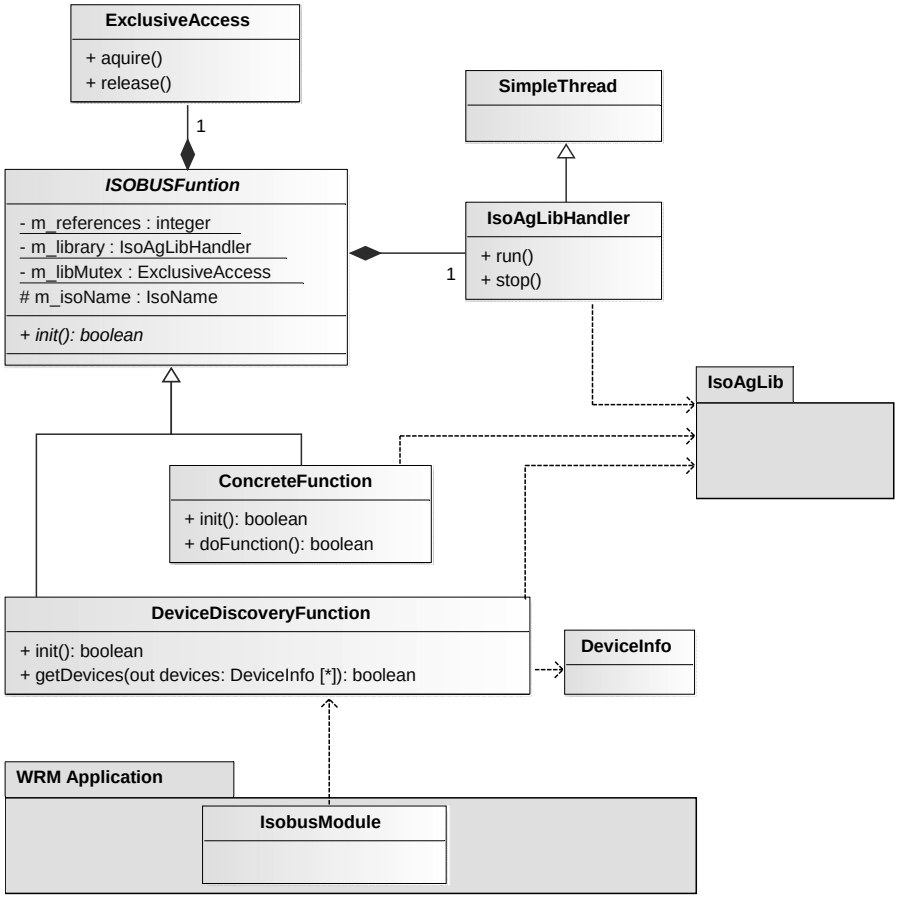

Figure 8.4: Class diagram: integrating ISOAgLib to WRM application.

In the WRM system the data sources that are available via ISOBUS are grouped to separate functions, such as device discovery. This way they can be easily integrated to the existing data model. The class diagram shows an abstract base class for these functions (*ISOBUSFunction*) as well as two example concrete functions that are inherited from it (*ConcreteFunction* and *DeviceDiscoveryFunction*).

The base class encapsulates common initializations that are required for all the functions that use the ISOAgLib. Such initialization is, for instance, registering the functions ISOBUS NAME, and performing an address claim with it. The implementation makes it possible that the same physical WRM device claims multiple logical bus addresses using different NAMEs if its required. By encapsulating the common parts to an abstract class, it is easier to add new ISOBUS related functionalities in future.

Another important responsibility handled by the *ISOBUSFunction* is the life cycle management of the ISOAgLib protocol stack. As mentioned earlier, the protocol stack should not be consuming resources when it is not used. This is why the *ISOBUSFunction* maintains a class variable reference counter which is the class in-

stances using the ISOAgLib. This way the IsoAgLib can be started and initialized when the first *ISOBUSFunction*-based instance is created and shut down when the last instance is destroyed. During the initialization a separate thread (*IsoAgLibHandler*) is started to feed the ISOAgLib's internal scheduler. The implementation also needs to be thread-safe so mutexes are used for mutual exclusion. All this is implemented in the base class, so the individual ISOBUS functions (childs of *ISOBUSFunction*) can assume that the protocol stack is up and running. The usage of the device discovery functionality is described in the figure 8.5.

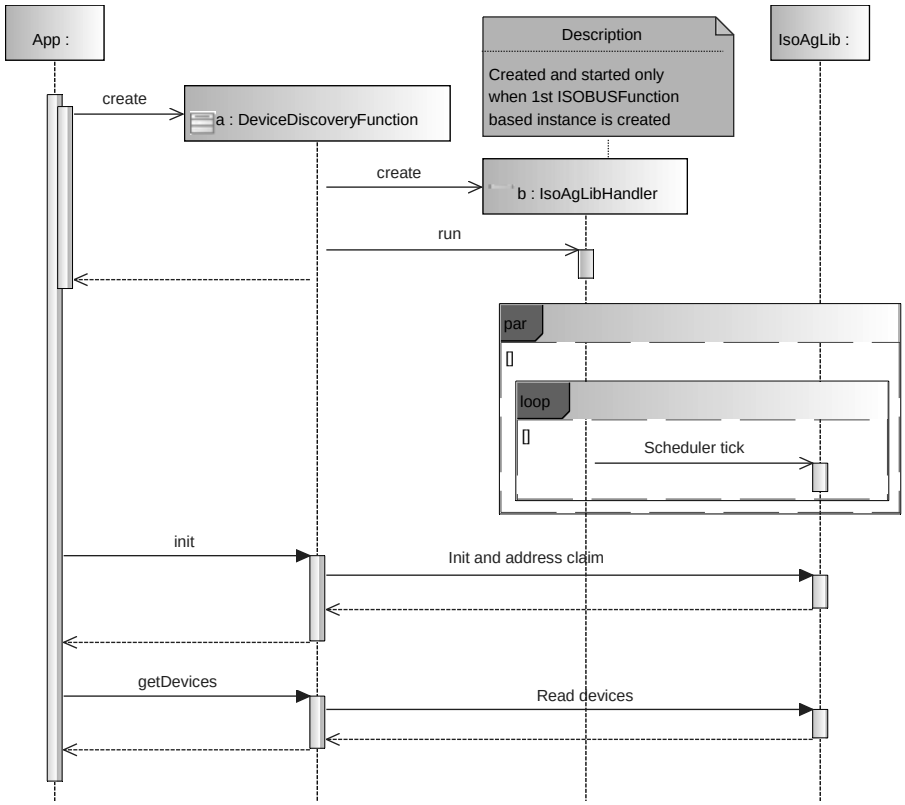

Figure 8.5: Sequence diagram: using the device discovery.

As the sequence diagram shows, using the ISOBUS related functionalities from the main application is very simple. This is achieved because all the protocol stack related operations have been abstracted from the rest of the application. The application can simply create a *DeviceDiscoveryFunction* instance, initialize it and start using it through its interface. Internally, the creation launches the ISOAgLib protocol stack if it was not already running. When looking at the sequence diagram above, it is worth remembering that the creation and start-up of the *IsoAgLibHandler* is handled by the *ISOBUSFunction* base class from which the *DeviceDiscoveryFunction* is inherited.

8.4 Testing

An important aspect in the software development is the testing. The diagram in the figure 8.6 shows the arrangement that was used for both development and local testing throughout the development.

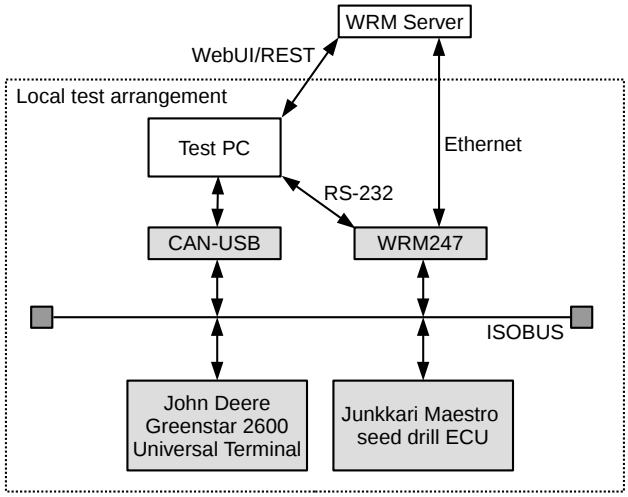

Figure 8.6: Local test setup.

The local test arrangement consists of CAN bus, ISOBUS devices and the test PC. In addition to the WRM247 remote device, John Deere Greenstar display as well as Junkkari seed drill implement ECU were also connected to the bus. The test PC was connected to the ISOBUS using a CAN-USB adapter which allowed to monitor the CAN traffic. Test PC was also connected to the WRM247 device's debug port using RS-232 adapter. Both the test PC and the WRM remote device were also connected to Internet using Ethernet which allowed them to communicate with the WRM server.

During development the WRM software and the ISOAgLib protocol stack were compiled using debug options which allowed to monitor their execution state using the RS-232 debug port. Also, the actual CAN frames sent to the bus were monitored and recorded using Wapice CANrunner software on the PC. The Junkkari ECU and the John Deere display allowed to test the device discovery functionality because their ISOBUS NAMEs were known beforehand. The Junkkari ECU is also developed by Wapice which allowed it to be modified for testing purposes. The ECU was modified so that know diagnostics trouble codes could be activated using the ISOBUS user interface. This allowed to test and verify the DTC collection functionality. The collected data was verified in the remote device using the debug port and also remotely using the server REST interface.

The ISOBUS file server functionality in the WRM device was tested by implementing a file server client to the Junkkari ECU. This was a standalone test which did not use the WRM server because the file server was not integrated to the main WRM application.

The test PC was actually the same computer that was used for development which allowed fast prototyping and also fast unit tests throughout the implementation. During the Agromassi project, these functionalities were also tested in co-operation with MTT/Vakola ISOBUS laboratory. The test arrangement used in these tests is shown in the figure 8.7.

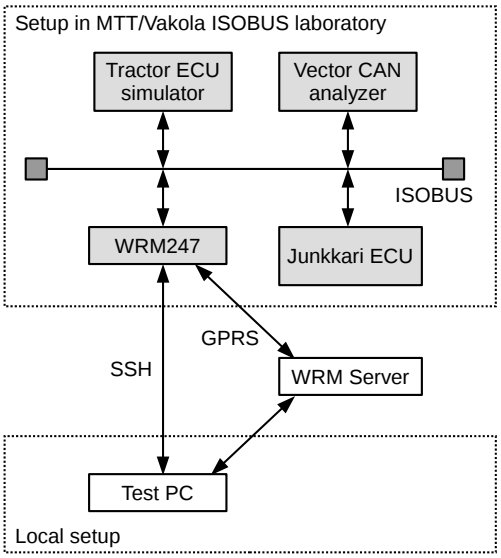

Figure 8.7: Remote connection test setup.

The WRM247 device was installed in the ISOBUS system in the MTT's laboratory. There were also a Junkkari implement ECU, a tractor ECU simulator and a Vector CAN analyser connected to the system. The WRM247 device used GPRS connection to communicate with the WRM server. The GPRS connection also allowed to directly connect to the remote WRM device using SSH (secure Shell). The tractor ECU simulator was configured to produce a known set of broadcast messages which were then collected using the WRM-device. The data collected was then verified using the WRM-server REST interface.

8.5 Results

The ISOBUS functionalities developed in this thesis and during the Agromassi project has been taken into use in the WRM system, and ISOBUS is now available as one of the protocol options. The ISOBUS support in general was implemented using a full ISOBUS protocol stack which in practice means that new ISOBUS features

can be developed quickly in future. This provides a good starting point for instance for customer customization. Furthermore, even though the file server functionality was not finished, the current implementation can be finalized with relatively little effort if it is needed in future.

Wapice also develops an ISOBUS implement ECU, and the general experience and the new knowledge of the ISO 11783 standard series will also benefit the ECU development. Furthermore, Wapice is participating in research projects which are related to ISOBUS technology, in which both the knowledge and the now available ISOBUS support of the WRM system can be utilized. For instance, CLAFIS EU project is one of those projects.

To further develop the ISOBUS support in WRM system, new functionalities could be added to the remote device's software. For instance, the process data logging interface would allow to monitor the actual implement variables which are also controlled by a task controller. The file server implementation would allow to provide files, firmware updates or tasks, for instance, to other devices on the bus.

9. CONCLUSIONS

The main objective of this thesis was to study the available telematics information and interfaces in an ISOBUS system, and also the applications this information enables. Furthermore, the architecture of the telematics system as well as implementation using the Wapice Remote Management system was also presented.

The main motivation for such telematics system is to provide a means to access agricultural machines in their working environment. This remote access makes it possible to implement value-adding services that can extend existing agricultural products, such as implements, and open new business opportunities. The telematics system could be used, for instance, for online fault diagnostics or performance monitoring which provides additional value to customers either directly or indirectly. Requirements for these application vary, and some will need real-time access to the machines whereas others will need data collected over a longer period of time.

An important aspect about the telematics system and it's implementation, which was studied in this thesis, is the information available in an ISOBUS system. The ISO 11783 standard, in which the ISOBUS protocol is based on, does not address remote telematics so existing interfaces were studied to find suitable interfaces that could be applied to telematics purposes. ISO 11783 provides methods for proprietary communication, so in theory it is possible to transfer arbitrary telematics information. However, when implementing a manufacturer independent generic telematics system, the proprietary messaging should be avoided because it limits the system usage to specific manufacturers.

There are, however, also many information sources that can be used in a standardized manner in an ISOBUS system. For instance, the tractor ECU broadcasts information about the tractor's state. The ISO 11783 also defines diagnostics functionalities which makes it possible to get information about diagnostics trouble codes and also query ECU information. Additionally, the standard also specifies methods for process data logging and file server functionality. So in short, the ISO 11783 provides standard interfaces to gather both telematics and process data, and to allow bi-directional communication with devices on bus and external systems.

To collect this data, a device that connects to the physical bus in the machine is needed. This device also needs to be able to communicate in the bus so it's software needs to support ISOBUS. The telematics system also requires a server subsystem

that aggregates data from multiple devices and provides both software- and user interfaces to access the system. The remote embedded devices use wireless networks to communicate with the server system.

In the telematics system the ISOBUS is only really visible in the low-level data collection devices that actually communicates in the machine bus. This means that when implemented correctly, the rest of the system, such as server and server interfaces, can be generic and thus easily re-used. This also means the ISOBUS telematics functionalities can be added to existing systems. This is exactly what was done in the WRM system during the Agromassi project. The ISOBUS support was added to the low-level remote management devices, and for the rest of the system the ISOBUS was yet another data source and they were used as is.

When the ISOBUS support was added to the WRM system, a pre-existing protocol stack was used. This design decision meant that once the protocol stack was ported to the WRM embedded platform, all of it's functionalities became available for application developers. Furthermore, this means that adding new ISOBUS related functionalities is easier because the protocol stack is already available. To further develop the WRM ISOBUS functionalities, for example, file server interfaces or process data logging interface could be added to the application.

REFERENCES

[1] Jansson, K., Thoben, K.D., The Extended Products Paradigm, An Introduction. In: Arai, E., Kimura, F., Goossenaerts, J., Shirase, K. Knowledge and Skill Chains in Engineering and Manufacturing. Vol. 168. Osaka, Japan 2002, Springer US. pp. 39-47.

[2] Kaloxylos, A., Eigenmann, R., Teye, F., Politopoulou, Z., Wolfert, S., Shrank, C., Dillinger, M., Lampropoulou, I., Antoniou, E., Pesonen, L., Nicole, H., Thomas, F., Alonistioti, N., Kormentzas, G., Farm management systems and the Future Internet era. In: Computers and Electronics in Agriculture. Vol. 89. Amsterdam, Netherlands 2012, Elsevier B.V. pp. 130-144.

[3] Robbemond, R., Kruize, J.W., Data standards used for data-exchange of FMIS (44). 2012, Wageningen University. 75 p.

[4] CC-ISOBUS Website. [WWW]. Competence Center ISOBUS e.V.. [cited: 4.4.2014]. Available: http://www.cc-isobus.org/en/association

[5] ISOBUS Compatible systems. [WWW]. DLG e.V.. [cited: 4.4.2014]. Available: http://www.dlg.org/isobus_en.html?&L=51

[6] Agrix project website. [WWW]. Aalto University. [cited: 3.4.2014]. Available: http://autsys.aalto.fi/en/Agrix

[7] Farmix project website. [WWW]. Aalto University. [cited: 3.4.2014]. Available: http://autsys.aalto.fi/en/Farmix

[8] Project Details - Assisting and adaptive agricultural machine - Agromassi. [WWW]. Agrifood Research Finland MTT. [cited: 3.4.2014]. Available: https://portal.mtt.fi/portal/page/portal/mtt_en/research/projectdatabase/Projectdetail?p_kielikoodi=GB&p_hakutyypi=perus&p_hanke_seqno=296041

[9] FutureFarm project website. [WWW]. FutureFarm Consortium. [cited: 4.4.2014]. Available: http://www.futurefarm.eu/about

[10] SmartAgriFood project website. [WWW]. SmartAgriFood Consortium. [cited: 4.4.2014]. Available: http://www.smartagrifood.eu

[11] CLAFIS project website. [WWW]. CLAFIS Consortium. [cited: 4.4.2014]. Available: http://www.clafis-project.eu/

[12] Sørensen, C.G., Fountas, S., Nash, E., Pesonen, L., Bochtis, D., Pedersen, S.M., Basso, B., Blackmore, S.B., Conceptual model of a future farm management information system, In: Computers and Electronics in Agriculture. Vol. 72. Amsterdam, Netherlands 2010, Elsevier B.V. pp. 37-47

[13] Lehmann, R. J., Reiche, R., Schiefer, G., Deliverable D100.1 - Review of the Literature and Future Internet Research. 2011, SmartAgriFood Consortium. 212 p.

[14] Miettinen, M., Oksanen, T., Öhman, M., Visala, A., Implementation of ISO 11783 Compatible Task Controller. 2006, Helsinki University of Technology. 6 p.

[15] Isobus-lisavarustepaketti N-, T-, M- ja S-sarjan traktoreihin. [WWW]. Valtra Oy Ab. [cited: 13.12.2013]. Available: http://www.valtra.fi/news/press/2005/659.asp

[16] Think ISOBUS. [WWW]. AEF. [cited: 5.3.2014]. Available: http://www.aef-online.org/fileadmin/MEDIA/downloads/com-pack/AEF_Broschure_EN.pdf

[17] Ohjainlaitteet. [WWW]. Junkkari Oy. [cited: 13.12.2013]. Available: http://www.junkkari.fi/kylvokoneet/ohjainlaitteet

[18] J1939-based. [WWW]. CAN in Automation e.V.. [cited: 24.11.2013]. Available: http://www.can-cia.org/index.php?id=19

[19] Controller Area Network (CAN). [WWW]. CiA. [cited: 18.9.2013]. Available: http://can-cia.org/index.php?id=systemdesign-can

[20] Introduction to SAE J1939. [WWW]. Kvaser AB. [cited: 1.11.2013]. Available: http://www.kvaser.com/en/about-can/higher-layer-protocols/36.html

[21] Junger, M., Introduction to J1939. 2010, Vector Informatik GmbH. 11 p.

[22] J1939-71 - Vehicle Application Layer. Warrendale, USA. 2003. SAE International. 379 p.

[23] What is ISOBUS?. [WWW]. AEF. [cited: 15.9.2013]. Available: http://www.aef-online.org/en/aef-projects/isobus/what-is-isobus.html

[24] ISO 11783-1:2007. Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 1: General standard for mobile data communication. Geneva, Switzerland. 2007. ISO. 94 p.

[25] Standards catalogue - ISO/TC 23/SC 19 - Agricultural electronics. [WWW]. ISO. [cited: 28.1.2014]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_tc_browse.htm?commid=47156&published=on

[26] Peterson, L.L., Davie, B.S., Computer Networks ISE : A Systems Approach. Fourth Edition. 2007, Morgan Kaufmann. 848 p.

[27] ISO 11783-3:2007. Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 3: Data link layer. Geneva, Switzerland. 2007. ISO. 42 p.

[28] ISO 11783-7:2009. Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 7: Implement messages application layer. Geneva, Switzerland. 2009. ISO. 144 p.

[29] ISOBUS Data Dictionary. [WWW]. VDMA e.V. [cited: 13.12.2013]. Available: http://dictionary.isobus.net/isobus/

[30] AEF ISOBUS Functionalities. [WWW]. AEF. [cited: 28.1.2014]. Available: http://www.aef-online.org/en/about-isobus/aef-functionalities/aef-isobus-functionalities.html

[31] ISO 11783-13:2011. Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 13: File Server. Geneva, Switzerland. 2011. ISO. 54 p.

[32] ISO 11783-14:2013. Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 14: Sequence control. Geneva, Switzerland. 2013. ISO. 65 p.

[33] Spitzer, S., NMEA 2000 Past, Present and Future. 2009. St.Petersburg, Florida, National Marine Electronics Association. 25 p.

[34] CANOpen. [WWW]. CiA. [cited: 5.3.2014]. Available: http://www.can-cia.org/index.php?id=canopen

[35] Pfeiffer, O., Ayre, A., Keydel, C., Embedded Networking with CAN and CANopen. 2003, RTC Books. 350 p.

[36] DeviceNET: Technical Overview. 2004. Michigan, USA, Open DeviceNET Vendor Association (ODVA). 8 p.

[37] Lennartsson, K., Fredriksson, L-B. SDS, DeviceNet and CAN Kingdom. [WWW]. Kvaser AB. [cited: 11.11.2013]. Available: http://www.kvaser.com/zh/about-can/higher-layer-protocols/59.html

[38] Fredriksson, L-B. A CAN Kingdom Rev.3.01. 1995. Kinnahult, Sweden. Kvaser AB. 117 p.

[39] ISOBUS in Functionalities. [WWW]. AEF. [cited: 11.4.2014]. Available: http://www.aef-online.org/fileadmin/MEDIA/downloads/blaetterkatalog/en/index.html

[40] ISO 11783-12:2009. Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 12: Diagnostics Services. Geneva, Switzerland. 2009. ISO. 24 p.

[41] agroXML website. [WWW]. KTBL e.V. [cited: 5.3.2014]. Available: http://www.agroxml.de/

[42] Wapice Ltd. website. [WWW]. Wapice Ltd. [cited: 26.2.2014]. Available: http://w3.wapice.com/

[43] WRM-system's website. [WWW]. Wapice Ltd. [cited: 26.2.2014]. Available: http://www.wrm.fi/en/

[44] WRM247+ datasheet. [WWW]. Wapice Ltd. [cited: 14.12.2013]. Available: http://www.wrm.fi/images/WRM247_DataSheet_2013.pdf

[45] WRM Server REST API, 2014, Wapice Ltd., 30 p.

[46] ISOAgLib website. [WWW]. OSB AG. [cited: 5.3.2014]. Available: http://www.isoaglib.com/en

[47] Junnila, S., Pajula R., Shroff, M., Siuruainen, S., Kwitek, M., Tuominen, P. Design of High-Performance CAN Driver Architecture for Embedded Linux. 13th international CAN Conference Part 5, Hamback Castle, Germany, March, 2012. Germany, CAN in Automation, 2012. pp. 1-9

[48] ISOAgLib System Architecture. [WWW]. OSB AG. [cited: 11.3.2014]. Available: http://www.isoaglib.com/en/function/systemarchitecture

A. WRM DATA MODEL

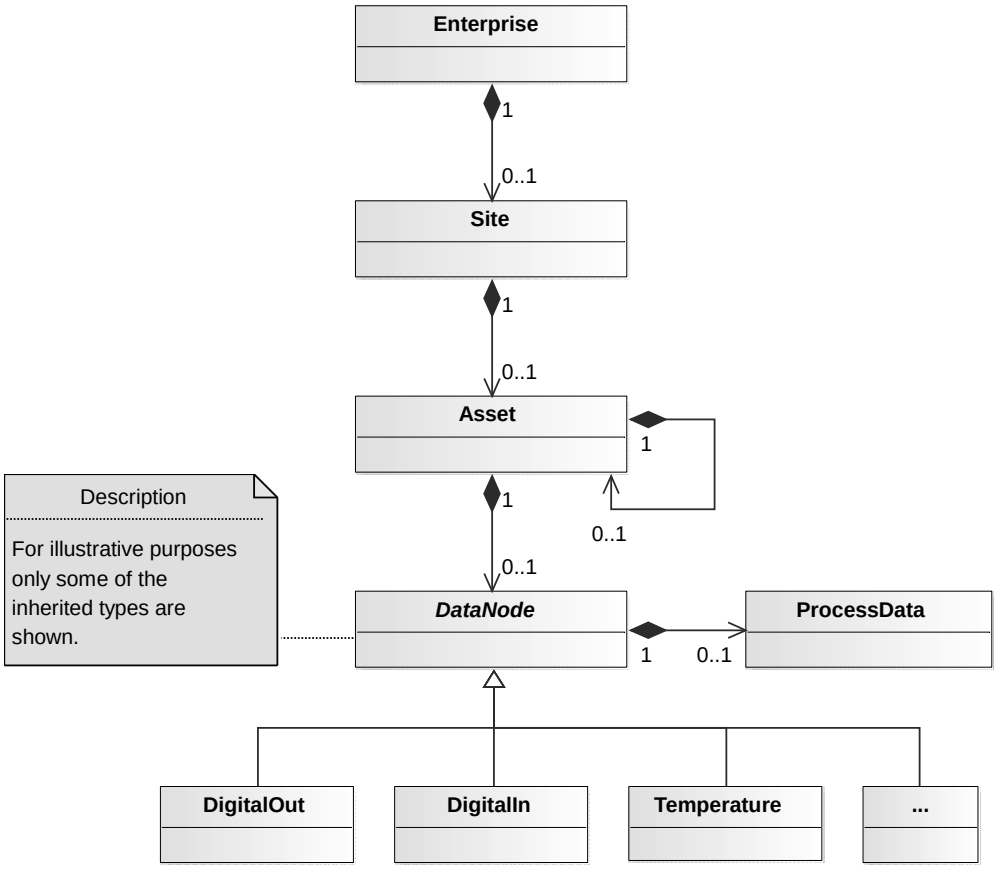

Figure A.1: Data model in WRM system.

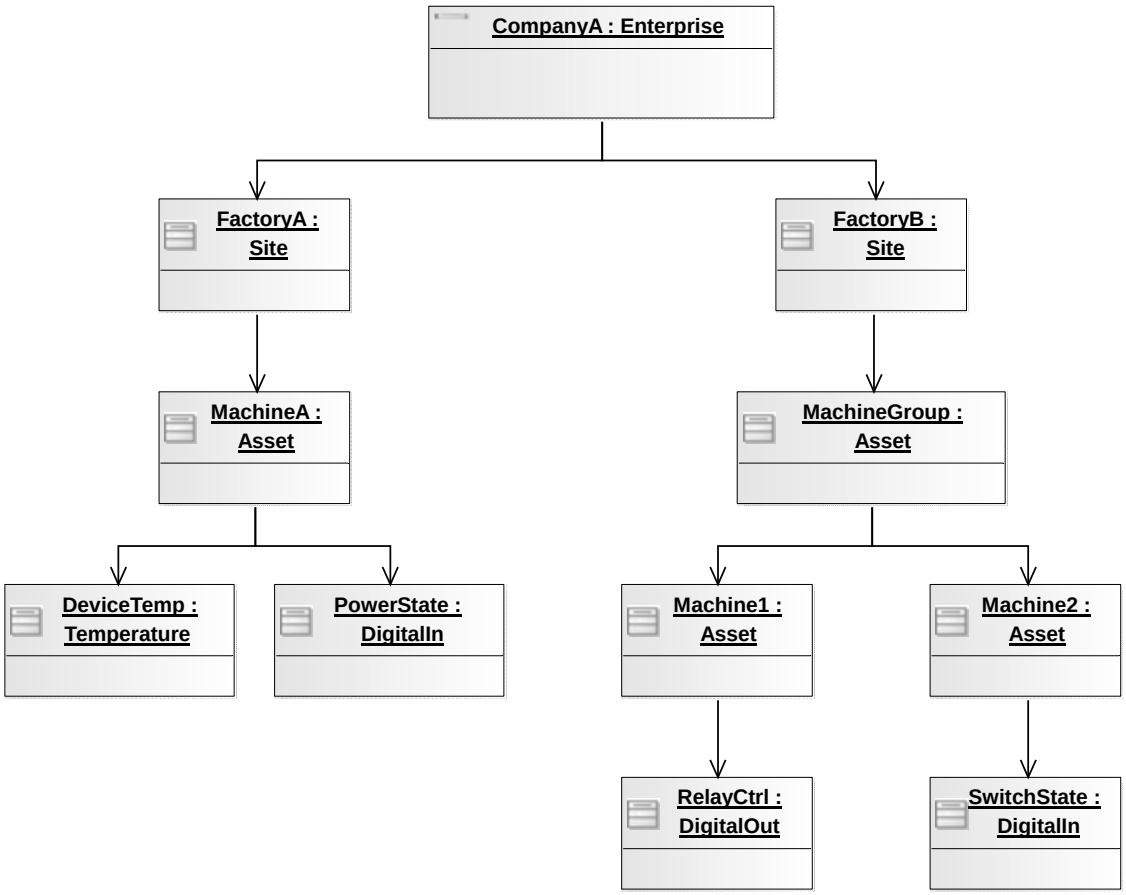

Figure A.2: Example enterprise object diagram.

B. ISOAGLIB HAL CLASS DIAGRAM

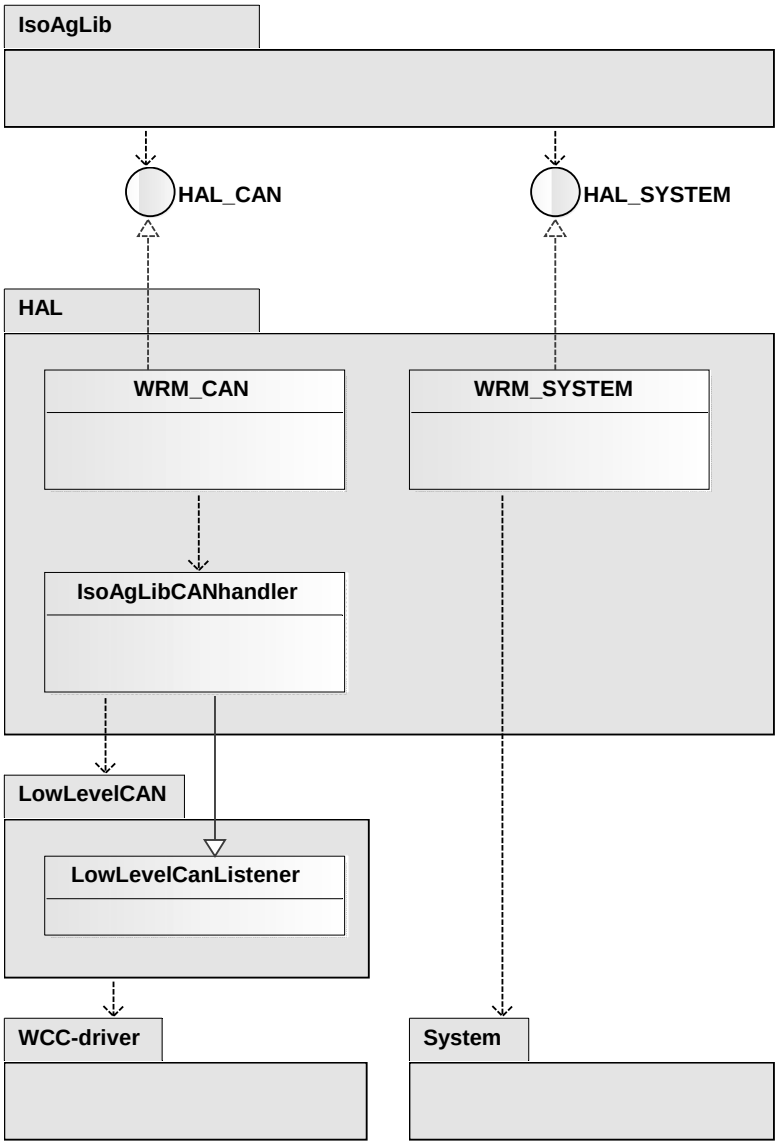

Figure B.1: Overview of ISOAgLib HAL implementation in WRM.